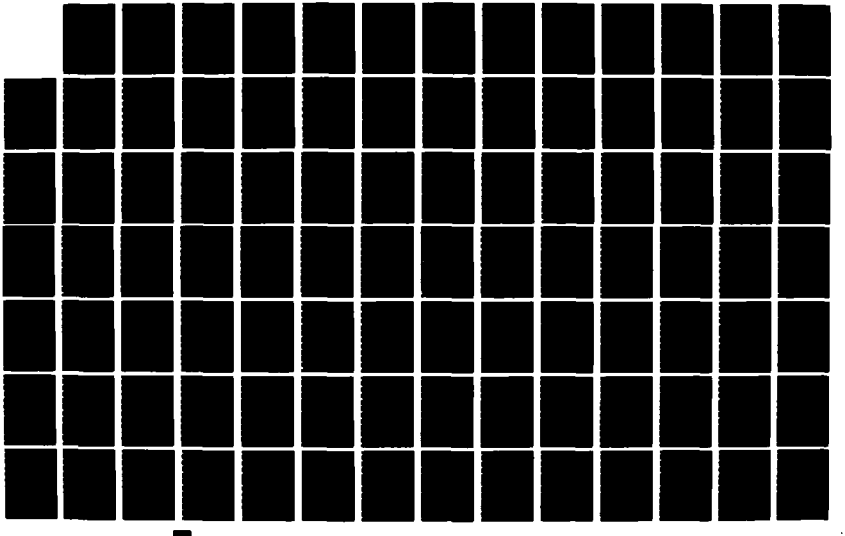
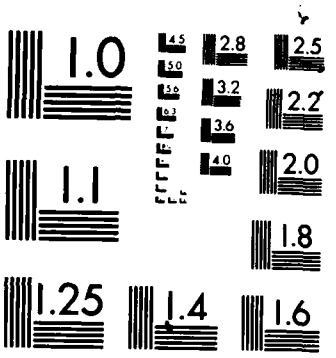
AD-A172 787 ANALYSIS AND DESIGN OF A MULTI-SECURITY LEVEL DISTRIBUTED DATABASE SYSTEM(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING 1/3

UNCLASSIFIED R A MOELLER DEC 85 AFIT/GCS/ENG/85D-10 F/G 9/2 NL





ANALYSIS AND DESIGN OF A MULTI-SECURITY LEVEL DISTRIBUTED DATABASE SYSTEM

THESIS

Ronald A. Moeller
Captain, USAF

AFIT/GCS/ENG/85D-10

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCS/ENG/85D-10

ANALYSIS AND DESIGN OF A MULTI-SECURITY LEVEL DISTRIBUTED DATABASE SYSTEM

THESIS

Ronald A. Moeller
Captain, USAF

AFIT/GCS/ENG/85D-10

Approved for public release; distribution unlimited

AFIT/GCS/ENG/85D-10

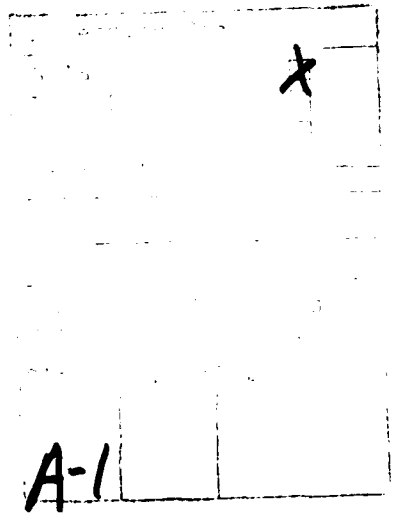
ANALYSIS AND DESIGN OF A MULTI-SECURITY LEVEL DISTRIBUTED DATABASE SYSTEM

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Computer Systems

Ronald A. Moeller, B.S.
Captain, USAF

December 1985



Approved for public release; distribution unlimited

Table of Contents

The following additional thesis volume is maintained at AFIT/ENG:

Volume II: Program Listings

List of Figures

List of Tables

ABSTRACT

A multi-security level distributed database system was designed based on database parameters and system requirements provided by Headquarters, Space Division. As the primary step in the analysis of this problem, a thorough investigation into the current state of the art of software verification techniques was made in order to determine exactly what a computer system's software and hardware could be "trusted" to perform correctly.

Furthermore, a selection was made from available secure local area network alternatives which would yield a solution that would be operational in other than a "system high" mode. The system chosen is currently being researched at the University of Newcastle upon Tyne in Great Britain. This approach involves locating a single security partition on a system which is physically and logically separated from the rest of the network. This separation is performed by a number of software and hardware mechanisms which can be formally proven correct.

Once the distributed secure system design had been suitably tailored for this application, a partial implementation of the design was successfully accomplished upon a local area network being developed at the Air Force Institute of Technology. The test results support the feasibility of this approach to the multi-security level distributed database problem.

ANALYSIS AND DESIGN OF A MULTI-SECURITY LEVEL DISTRIBUTED DATABASE

I. Introduction

Background

Over the past twenty years, computer usage has grown to the point that it influences almost every aspect of our lives. Concurrent with this growth has been the development of such system users' needs as the need to share system resources, the need to more effectively utilize system components, and the need for intercomputer communication. This need for the controlled sharing of system resources has grown, not only in the number of people involved, but in the geographic dispersion of both people and resources and their need for increasingly fast access to and transfer of such information. This tremendous expansion has presented new technological problems in many areas, but particularly in that of computer system security.

Recent efforts to build secure computer systems have resulted in limited success. Secure system models have been proposed and several systems have been implemented based on these models. The primary objective of each of these systems is to ensure that a system user has access only to such information for which he is both cleared and has a need to know, an objective which must be proven attainable. However, it is the verification requirements placed on these systems which prevent them from fully achieving their

primary goal. Currently available verification techniques simply cannot provide adequate verification assurances for large software programs such as a computer operating system or a database management system. So as the need for inter-system communication becomes stronger and the number of such systems increase, solutions to the security problems will be sought even if such solutions provide only temporary relief.

Currently, the four modes of secure computer operation within the Department of Defense are dedicated, system-high, controlled, and multilevel, with each mode providing a different level of system flexibility (11:86). Perhaps the most inflexible mode, yet the mode with the greatest degree of trustworthiness, is the dedicated mode. Systems operating in the dedicated mode process data at one specific security level at any given time. If jobs with both a higher priority and a more sensitive clearance are presented to an installation for execution, the computer system must be completely sanitized, both before and after the new job is allowed to run. The second mode of operation, system-high, provides relief to the previous mode's obvious limitations. Within this mode of operation jobs of differing levels of sensitivity may coexist on the same system, thereby eliminating the time consuming sanitation requirements. However, all system outputs are classified according to the most sensitive information on the system. At some installations, each user is provided the opportunity to downgrade his own output, an often tedious and potentially

error-prone operation.

The final two modes of operation are the most flexible. Systems operating in the controlled mode are authorized to process information at up to three levels of classification concurrently. Furthermore, systems operating in the controlled mode are not required to classify all output according to the highest classification of the data maintained by the system. This ability to distinctly classify its output is given to controlled mode systems because of the additional assurances provided by verifiably secure software. The multilevel mode of operation provides for concurrent processing of any number of levels or categories of classified information and the system provides a high level of assurance that all information of the various levels will be segregated during processing. Additionally, the multilevel mode must guarantee that information provided to system users is only information for which the user has been properly cleared. It is one of the primary objectives of this study to examine the degree of trustworthiness that multilevel mode systems currently provide.

At this time, it seems appropriate to define terms used throughout this study to provide a common understanding of what this project will attempt to accomplish. First, a database is an organized collection of data stored, more or less permanently, in a computer. A distributed database is a database which is kept at dispersed locations with two or

more computers controlling access to the different parts of the database and several computers may be used to interface with users, who may be at different locations from the data itself. A computer network provides the necessary communication links between the computers on which the database resides. Finally, security, in both a network and a database sense, involves three basic aspects of protection. These aspects include providing controlled access to resources, providing controlled use of those resources, and providing assurance that the desired level of protection is maintained throughout the system (12:6). With these definitions, the following problem statement should become clearer.

The analysis and design of a multi-security level distributed database entails the resolution of three distinct, yet related, problems. The first problem is to define the topology of the computer network upon which the database will reside. Specifically, existing computer networks, along with their security characteristics, will be examined to determine an optimal network configuration. Concurrent with the network examination will be an investigation of the methodologies available pertaining to the design of the database. This problem concerns the strategy to be used in dividing the database among the various nodes of the network. Again, the security aspects of the individual items within the database will play a crucial role in determining the partitioning strategy. The

final problem, once it has been decided how the computers within this local area network will be communicating with one another and how the database will be divided among the various network nodes, addresses internetwork communication. This area of the study will determine which node or nodes of the local area network will act as the interface, or gateway, to an external communications network.

Problem and Scope

The objective of this study is to propose a database design which will meet the specific requirements of the thesis' sponsor, Headquarters Space Division, Air Force Systems Command. These requirements include the actual database parameters and the operational environment in which the database will reside (For a complete description of these requirements, see Chapter II). In other words, a comprehensive architecture will be defined for a specific database in a particular installation. Furthermore, the study will be limited by the currently available hardware and software or some variation thereof and the methodologies in use today to verify the degree of trustworthiness each system component may assume.

Once the design has been accomplished, its implementation will be attempted, using existing Air Force Institute of Technology (AFIT) facilities. The AFIT Digital Engineering Lab's (DEL) local area network and distributed database management system will provide a good, although

limited, environment to implement the design which will be proposed by this study. The DEL facilities are limited to a single local area network architecture with the majority of the network nodes being microprocessors using single task operating systems (9:5). Furthermore, the distributed database management system is the result of a recently completed thesis and provides limited query capabilities (3:73). However, limitations aside, the DEL facilities will provide a good hands-on experience for the implementation of the design to be proposed by this study.

Assumptions

The outcome of this project will be based substantially on the following assumptions:

1. The physical environment is secure; that is, system operations are performed in physically hardened or guarded facilities.
2. All personnel having authorized access to the system have been cleared through appropriate background investigations to handle classified information.
3. No unauthorized, intentional penetration attempts into the system will be made; in other words, a nonmalicious environment exists within the local area network.
4. The external communication network's transmission lines are physically secure.
5. The program to implement the recommendations of this study will be well funded but will not have unlimited economic resources.

The above assumptions are based upon the problem specification provided by, and through subsequent conversations with, the thesis sponsor. These assumptions direct the study to specific security related areas, primarily software

security, while avoiding other security areas not considered important at this time. Assuming certain system safeguards, particularly a nonmalicious environment, emphasizes the specific direction this project will take and limits the types of potentially compromising situations which may occur. One of the goals of this study is to prohibit situations in which a user of the system unintentionally has access to information for which he is neither cleared nor has a need to know.

Approach

Deriving the final database design involved several interrelated steps. The first step was an extensive literature search to examine alternative designs of secure local area networks. During this phase, the study defined critical issues and problems that related to network security, described the various mechanisms implemented in a particular security policy, discussed the tradeoffs which related to these mechanisms, and finally, selected a particular architecture which both met the installation requirements provided by the thesis sponsor and provided the most trusted system available today. A related issue concerned the manner in which the local area network interfaced with the external communications network. This area is of particular importance due to the fact that the interface must be one of the most secure components of the entire design.

Once the network architecture alternatives had been investigated, the study addressed the distributed database design issues. Integral to the examination of multi-security level distributed database design issues was the degree of trustworthiness existing operating systems may assume. Therefore, before resolving the database design problem, this study must address not only the present state of secure database management systems but also the unresolved issues in secure operating systems. Finally, given either a complete or a partial compatibility between the proposed design and existing facilities within the DEL, an implementation of this design was attempted. In the case of only partial compatibility, all variations between the ideal design proposed and the system implemented were fully annotated in this study along with the associated simplifications these modifications entailed.

Sequence of Presentation

The remainder of this thesis is presented in six chapters. Chapter II describes the system requirements, specifically, the database parameters and the secure operating environment in which the database resides. This chapter also addresses the overall installation requirements, particularly the message traffic flowing into and out of the local area network. The third chapter discusses the present state of computer software verification methodologies as related to secure computer operating systems and secure database management systems. Chapter IV examines

alternative secure network designs and proposes a local area network architecture suitable for the required operating environment. The design of the database is presented in chapter V, a design which is almost totally dependent on the information presented in the preceding chapters. The sixth chapter includes all the details of the attempted implementation of the database on the digital engineering lab's local area network. The final chapter presents a design summary which includes all conclusions and recommendations generated by this study.

II. System Requirements

Introduction

The proposed distributed database design must satisfy both the network and database requirements as presented by the thesis sponsor. These requirements describe the nature of the environment in which the final product will be operating. More importantly, the requirements provide the author an excellent starting point from which his research could proceed, particularly in the system security aspects of the thesis problem.

Network Requirements

The organization requiring the development of this system, known from this point on simply as the user, has a need for a multi-security level distributed computer system to aid in the planning of aircraft flight paths. This distributed computer system, or local area network, consists of a minimum of three computers, with security levels of the data contained therein ranging from level zero as the most restrictive to level two as the least. The local area network will be connected to a second network, an external communications network, through which queries and updates to the aircraft flight path database are made. These queries and updates are in the form of messages pertaining to an aircraft's planned flight path and associated weather and surface-to-air missile threats.

Computer A{L}, the first of the three primary hosts,

maintains the weather related portion of the database. (For the remainder of this study, whenever one of the principal database computers is mentioned, it will have either a "H", "M", or "L" in braces appended to its single letter identifier. The "H" indicates the computer maintains level zero information, the "M" indicates level one information and the "L" indicates level two information.) The weather data is classified at level two and is received via messages either directly from the external communications network or through an intermediary processor within the local area network. The weather messages describe actual weather conditions within geographical regions by altitude and time. Weather related queries are sent to computer A{L} through the local area network primarily by computer C{H} and involve requests for weather conditions for a specific flight path. Computer A{L} calculates what region(s) the flight path covers and provides computer C{H} all associated weather information.

The second primary host, computer B{M}, maintains the surface-to-air missile (SAM) portion of the database. The SAM data is classified at level one and is received similarly to the weather data. However, SAM data is received in two different message formats. The first message format provides a SAM site's operational status and contains such information as location, status, and missile type. The second SAM message format describes the characteristics of a particular type of SAM and includes both

altitude and range parameters. SAM queries are sent by computer C{H} to computer B{M} and involve requests for the hostile regions a planned flight path covers. Computer B{M} analyzes all legs of the flight path to determine both "free" and "kill" zones. If the flight path covers only "free" zones, computer B{M}'s response is simply "NO DANGER". Otherwise, a "DANGER" message is sent along with the number of SAMs within the zone and all associated site and missile data.

Computer C{H}, perhaps the most important of the three primary hosts, maintains the flight path portion of the database. The flight path data is classified at level zero and is received similarly to both the weather and SAM data above. Once computer C{H} has received a flight path, it queries both computers A{L} and B{M}. Based upon the information received back from the other primary hosts, computer C{H} generates an updated flight path to include the associated weather and SAM data. The updated flight path is then sent either directly to the user via the external communications network or to an intermediary processor in the local area network.

In addition to the network requirements described above, each computer has a terminal from which a user may sign on and query different parts of the database depending on which particular computer he is signed on. I all cases, the user is allowed to access only such information that he is not only cleared for but which is at or below the security level

of his computer. For example, a user logged on to computer B{M} may view both level two and level one information provided he has been properly cleared for such access. However, even if this same user is cleared to view level zero information, software security measures prevent such access since computer B{M} is only capable of processing up to level one data. Other facilities provided local users includes reviewing message logs, reviewing the actual messages themselves, and composition and transmission of messages through the external communications network.

The final requirement placed on the local area network is the creation and maintenance of an audit trail. This audit trail is required so that the network security officer will have the necessary information to reconstruct any activities which took place throughout the local area network. The network audit trail must be available for regular or on-demand security reviews of both the entire network or individual host activities. The audit trail must include, at a minimum, the following information:

1. Host identifiers, user identifiers, and session security level.
2. Identification of protected resources accessed during the session.
3. Session starting and ending times.
4. Any exceptional conditions occurring during a session.

As just introduced, the term session for this work will mean any network activity for a period of time involving communication both across the network between two active entities

or one entity actively querying system resources within a single host.

Database Requirements

The two most important requirements related to the design of the database are the multiple levels of security classifications of the data contained therein and the processing requirements placed on the various hosts of the local area network. The specific security classifications are maintained strictly at the file level; in other words, each record and the fields it consists of are classified according the classification level of the file in which they reside. Security classifications are also integral to the partitioning strategy discussed in Chapter IV. The processing requirements of each of the primary hosts mentioned above are such that the database must be distributed to provide necessary response times to user queries, as specified by the thesis sponsor. No analysis will be made of these response times or host workloads since the thesis problem is structured to be of a distributed nature.

The specific database parameters are generically grouped as weather, surface-to-air missile sites, surface-to-air missile types, planned flight paths, incoming and outgoing message logs, and audit trail. The weather portion of the database includes the following parameters:

1. Region Names - 10 characters maximum.
2. Northeast latitude - 90.000 to 90.000 degrees.

3. Northeast longitude - 0.000 to 360.000 degrees.

4. Southwest latitude - 90.000 to 90.000 degrees.

5. Southwest longitude - 0.000 to 360.000 degrees.

6. Weather conditions by time and altitude to include:

   a. Julian day - 0 to 366.

   b. Seconds since midnight - 0 to 86,400.

   c. Altitude - ground, 10,000,20,000,...,100,000.

   d. Wind speed - 0 to 200 knots.

   e. Wind direction - 0 to 360 degrees.

   f. Cloud cover - clear (C), partly cloudy (P), broken clouds (B), cloud cover (X).

   g. Precipitation - none (N), rain (R), snow (S), hail (H), thunderstorms (T).

Only the most current weather information will be maintained within the database, with updates being received approximately every six hours. Historical weather data is simply discarded.

As was mentioned above, the surface-to-air missile data is grouped as both sites and missile types. The sites parameters include:

1. Name - 10 characters maximum.

2. Missile name - 5 characters, SA-1 to SA-19.

3. Latitude - 90.000 to 90.000 degrees.

4. Longitude - 0.000 to 360.000 degrees.

5. Operational status - fully operational (O), under construction (U), closed (X), destroyed (D).

The missile type parameters include:

1. Name - 5 characters, SA-1 TO SA-19.

2. Minimum altitude - Ø to 1ØØ,ØØØ feet.

3. Maximum altitude - Ø to 1ØØ,ØØØ feet.

4. Missile range - Ø to 9ØØ,ØØØ feet.

Both SAM site and SAM type data are updated from messages received through the communications network. All historical missile data will be maintained even if a site is closed or a missile type becomes obsolete.

The most sensitive information maintained within the database is the flight path data which is classified at level zero. This information consists of the following parameters:

1. Name - 1Ø characters maximum.

2. Path leg - an array of up to 2Ø entries:

    a. Julian day - Ø to 366.

    b. Seconds since midnight - Ø to 86,4ØØ.

    c. Latitude - 9Ø.ØØØ to 9Ø.ØØØ degrees.

    d. Longitude - Ø.ØØØ to 36Ø.ØØØ degrees.

    e. Altitude - Ø to 1ØØ,ØØØ feet.

Once the flight path has been analyzed by both computers A{L} and B{M}, this information will be expanded to include the pertinent weather and SAM data.

The final three database groups together provide the information necessary to maintain a record of all the activities which have taken place both within the local area network and between the local area network and the external communications network. The first of these groupings, incoming message log, includes the following parameters:

1. Message identification - IYYNNNN, incoming (I), year (Y), number (N).

2. Originator - 10 characters maximum.

3. Date-Time-Group - DDHHMMSSZMMMYY.

4. Subject - 40 characters maximum.

5. Security level - level 0 to level 2.

The outgoing message log is identical to the incoming message log except for the Originator parameter. In place of this parameter, the outgoing message log has a To parameter, which is also a maximum of ten characters long. The audit trail group parameters include:

1. Host identifier - 2 characters maximum.

2. User identifier - 6 characters maximum.

3. Session identifier - 5 characters maximum.

4. Resource accessed - 11 characters maximum.

5. Session start time - 0 to 86,400 seconds.

6. Session end time - 0 to 86,400 seconds.

7. Exceptions - 40 characters maximum.

The Exceptions parameter provides a brief description of any irregularities occurring from the moment a logon is attempted until the user logs off the system. Examples of such irregularities includes errors made during the identification/authentication process or an attempt to access a file for which this particular user does not have access permission. Obviously, this is not an exhaustive list of error conditions which may occur. As was mentioned previously, these audit trail facilities enhance the network

security monitor's chances for reconstructing any unapproved activities by recording all network activities.

Summary

This chapter has provided a brief glimpse at a number of issues which are addressed in much greater detail throughout the remainder of the thesis. Perhaps the most important issue raised concerns the security classifications of the data. This single issue influences not only how the database will be partitioned among the local area network nodes but network architecture and network topology issues as well. The installation unique requirements were also presented in in this chapter along with the database parameters. The presentation of these requirements was a necessary prerequisite since the design of a comprehensive security architecture is almost totally dependent on each system's peculiarities. A better understanding of security policy, local area network design, and security mechanisms provides an awareness of the tradeoffs to be considered when designing distributed computer systems, particularly multi-security level distributed databases.

III. Secure Computer Systems

Introduction

In order to more fully understand the fundamental nature of the secure computer system problem, this chapter investigates several related topics beginning with an introduction to the most prominent secure operating system design technique, the security kernel. Integral to the security kernel presentation is a thorough discussion of the Bell and LaPadula model, which provides the security model upon which the security kernel approach is based (1:15). Subsequent to this discussion is a brief presentation of the Grohn model, which extends the Bell and LaPadula model to database management systems. Finally, a survey of existing computer software verification methodologies is presented

Security Kernel

For the past fifteen years, government and industry have investigated techniques for developing computer systems upon which a given set of security rules can be reliably enforced. The results of this intensive investigation have produced many different approaches to the design of secure computer systems. At present, the most widely used design strategy incorporates the concept of a security kernel (11:87), a concept which includes all the hardware and software mechanisms which enforce the authorized access relationships between the subjects (users) and objects of a computer system. The primary motivation behind the develop-

ment of a security kernel, from the Department of Defense's point of view, was to provide an environment in which multiple security levels of information could reside on a single computer system with a user only having access to such information as he was both cleared to use and had a need to know. Obviously, the objective of the security kernel approach is for the kernel to perform in an operational environment such that the separation of different levels of classified information is accomplished by the operating system and associated hardware. In other words, the security kernel provides the capability for a truly multilevel mode of operation. Whether or not such an approach does indeed meet this objective will be discussed in the final section of this chapter.

Before describing the specific principals which are important in a security kernel's design, it is appropriate to define more fully the relationship between a reference monitor and a security kernel. A reference monitor is a conceptual notion which involves the checking of each reference by a subject (user or program) to an object (file, device, user, or program) and determining whether the access is valid under the system's security policy (11:88). A security kernel consists of those software and hardware components of the computer system which implement the reference monitor concept and includes all relevant security software. As envisioned by the United States Air Force's Colonel Schell, who first introduced the idea of implemen-

ting the reference monitor concept with a security kernel in 1972, a security kernel was a "compact security 'kernel' of the operating system and supporting hardware such that an antagonist could provide the remainder of the system without compromising the protection provided (1:15)."

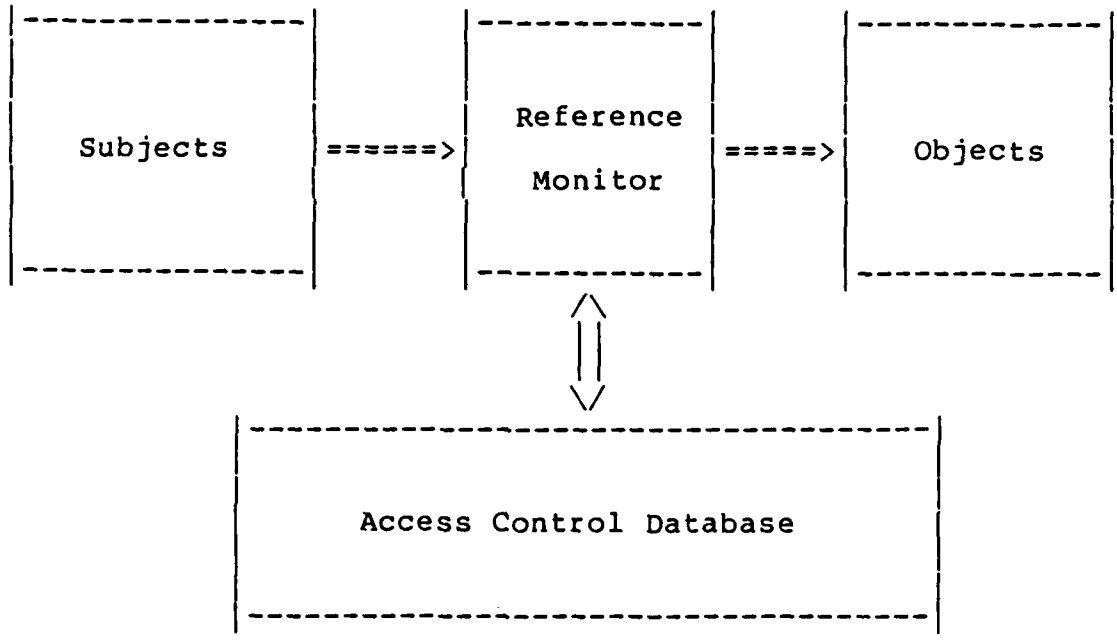


Figure 1. Reference Monitor

The two basic security kernel design principals are first, that a security model should be formally defined, and secondly, that the security model should be faithfully implemented. The formal definition of a security model is based upon a specific set of protection policies. This requirement is a distinguishing factor between a security kernel based system and other approaches to the design of secure operating systems. These other systems, such as

capability based machines, provide general purpose mechanisms, whereas the security kernel approach specifically addresses both policy and mechanisms (1:14).

The security policy aspect of the kernel approach is divided into two major categories, nondiscretionary and discretionary. Nondiscretionary policy involves security rules which are imposed on all system users while discretionary policy is the set of security rules specified arbitrarily by each user. In other words, the specification of the rules is discretionary in the sense that a subject with access permissions is capable of passing those permissions to any other subject. Both the nondiscretionary and discretionary policies are translated into a security model which is composed of a set of mathematical rules. The security model must define the information protection behavior of the system as a whole and include a "security theorem" to ensure the model's behavior consistently complies with the nondiscretionary and discretionary policies described above. The model enforced by most security kernel designs is the Bell and LaPadula model (2:15), which is described in in the next section.

The second security kernel design principal, faithful implementation, requires that a set of formal verification techniques be applied at each stage of the design and implementation process. The different classes of formal verification methodologies as applied in the kernel design process include techniques to prove that a kernel's intended

behavior is secure with respect to the policy model, techniques to verify a correspondence between the mappings of the various design and implementation specifications, and finally, techniques which simply show a correspondence between the kernel's specification and its implementation (10:17). Each of these different classes, along with specific verification techniques, is discussed more fully in a later section within this chapter.

Typically, an operating system is broken up into functional areas such as processor, memory, device, and information management. In designing a security kernel, each functional area must be analyzed to see if it performs any security related tasks. If so, these specific functions become candidates for inclusion within the kernel. Most importantly, any part of the operating system which manages resources shared by multiple users, such as memory, must be included within the kernel. As pointed out in the Ame's article, the design of a security kernel is really a balancing act among the various design issues, such as deciding which functions are security related, performance and functionality tradeoffs, and consideration for operating system complexities. The objectives in such a design environment are twofold. The kernel must be foolproof and it must not degrade system performance. Unfortunately, as will be discussed later, present systems do not effectively achieve this second objective.

There are four architectural areas in which specific

hardware and software mechanisms have proved useful or necessary to support kernel based operating systems. The first area concerns the notion of explicit processes. Each system user initiates one or more activities within the system which act on his behalf. These activities, or processes, must each be explicitly identified as belonging to a specific user so that each process may be associated with the user's specific access privileges to a particular object. The user's identification and access privileges must be nonforgeable. Directly related to the explicit process idea is the need for the security kernel to provide support for multiprogramming and interprocess communication. Since each reference request must interface directly with the kernel, and in a multiprogramming environment there are several processes making such requests simultaneously, it would be advantageous if all such requests could be handled through system hardware. With both context switching and interprocess communication being supported through hardware mechanisms, kernel simplicity and performance would substantially improve.

The second architecture area addresses memory protection issues. The most important kernel facility required is a mechanism for mediating memory accesses. This mechanism is achieved through a virtual memory system in which each logically distinct memory object has an associated descriptor with attributes such as size, access mode, and access class. These objects are referred to as segments and are

allowed to dynamically increase in size up to some system maximum. The security kernel's software manages a segment's descriptor, which in turn manages or controls the virtual address mapping hardware. Complete access mediation is provided since the hardware must interpret the relevant descriptor for each access to memory (1:19).

Perhaps the more novel security kernel architectural idea of the four is the concept of execution domains. The most common operating system structure consists of a two level hierarchy in which the supervisor and user domains are distinct. However, in the design of a security kernel operating system, the kernel also requires a distinct domain. Due to the unacceptability of including the supervisor mode with the user mode, it was proposed that a third mode be supported in a secure kernel environment. In order to ease the transition between execution domains, entry into the most privileged domain must be limited to a few well defined entry points. However, in an effort to design both a secure and a simplistic kernel, a hardware mechanism which supports multiple entry points, similar to a procedure call in that each kernel function has a distinct entry point, has usually been implemented.

The final architectural issue concerns the mediation of input/output (I/O). The resolution of kernel related I/O issues must address the two most frequently used I/O techniques in use today. The simplest of these two techniques is programmed I/O, synchronous transfer, in which the

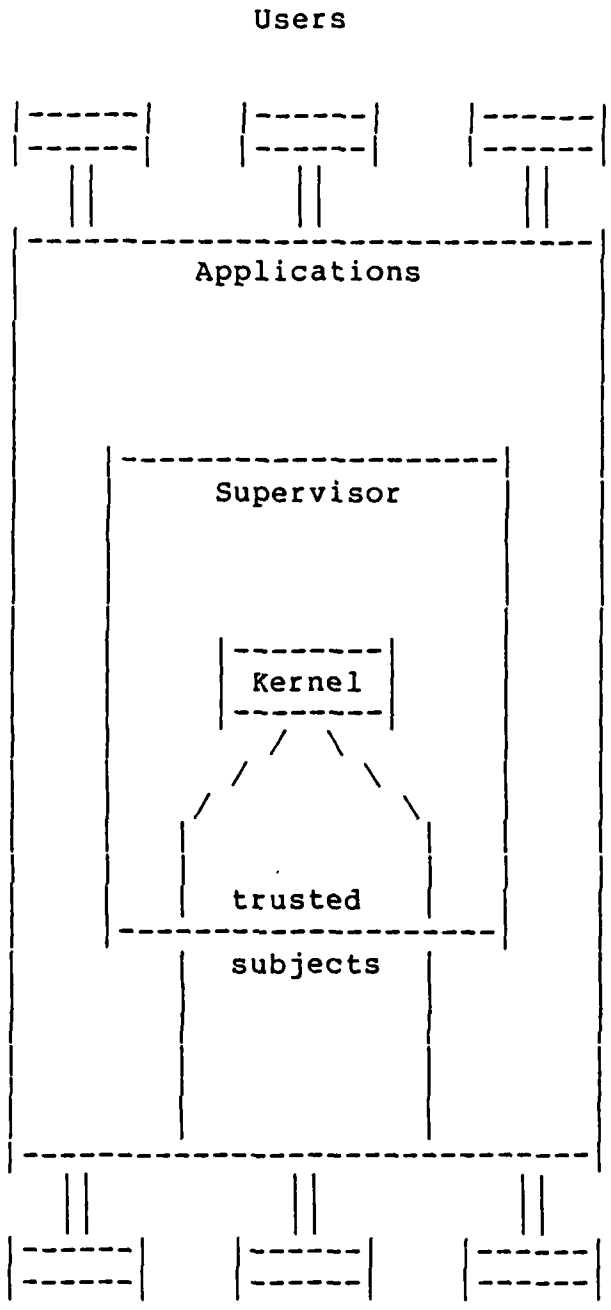


Figure 2. Kernel Structure

central processing unit (CPU) is instructed to transfer each byte between main memory and the I/O device itself. In a secure environment, the start I/O instruction would be maintained in the most limited domain, the kernel domain. Therefore, the user and supervisor domains must request I/O actions to be made by the kernel on their behalf. The second I/O technique involves the asynchronous transfer of information between memory and secondary storage. Usually this technique requires the CPU to notify an I/O processor when I/O is to be initiated. Once the I/O processor is awakened, it handles the transfer of information, only notifying the CPU when it terminates. In a secure environment, the I/O processor works entirely within the kernel's domain and uses physical memory addresses supplied by the kernel. This restriction of the I/O processor necessitates the inclusion within the kernel of the virtual memory system's translation algorithm.

As described above, the security kernel is a complex mechanism which performs many functions. In order to conceptually reduce this complexity and at the same time provide a means to effectively demonstrate the system's security, several formal models have been developed. Because of the difficulties of trying to capture the complexities of the real world in a formal structure, each model deviates from reality in certain respects. However, the different security models all base their interpretations of a computer system on the finite state machine model. The

finite state machine consists of a finite set of states in which a transition function provides a means to move from one state to another based upon the machine's current state and an input value. The transition between states may also generate some output values as determined by the transition function. The three most common computer security models include the lattice, the information flow, and the access matrix. Much has been written about these particular models, so in the interest of keeping this discussion as brief as possible, only a few comments are made regarding the access matrix model, which is the basis for the Bell and LaPadula model.

The access matrix model was developed as a generalized description of computer operating system protection mechanisms. The model depicts controls on users' access to information without regard to the semantics of the information in question. The particular mechanisms used in the model include a reference monitor and an access control matrix. However, when classified information is involved, the semantics of the information becomes very important. For this reason, models based on the access matrix have been extended to include rules concerning the classifications of the information maintained by the system. In systems based on this extended model, the protection of information is the responsibility of the information's owner. The owner can grant access to this information to any user, who, in turn, may distribute the information in any way he pleases. Such

a policy makes it very difficult to prove any theorems about the flow of information without specializing the model. However, the primary advantage of this model is that it neatly separates the mechanisms for enforcement from the policy enforced. The simplicity of the access matrix model makes it very appealing, particularly the way it defines subjects, objects, and access control mechanisms (10:257). Consequently, it has served as the basis for a number of other models, one of which is the topic of discussion in the next section.

The Bell and LaPadula Model

The Bell and LaPadula model (2:5-70), here after referred to as simply the model, consists of two major facets: a descriptive capability and general mechanisms. The model has the ability to abstractly represent the elements of a computer system which are relevant to the treatment of classified information. The computer system is viewed as a finite state machine in which a request for transition from one state to another generates an appropriate response (decision) based upon an associated rule (2:9). However, before discussing how these decisions are made, it is necessary to describe the model's more fundamental aspects.

The descriptive phase of the model begins by introducing the notion of subjects and objects, where a subject is considered an active entity (process) and an object is a passive entity (file). The model depicts the control of

access by subjects to a set of protected objects based upon some predetermined security policy. As defined in the Mitre Corporation report, the "system" is all sequences of three-tuples consisting of system states, requests (input), and decisions (outputs), along with an initial state, which satisfy some relation on successive states. Associated with each passive entity (object) are a set of access attributes which includes the following modes of operation:

1. e(execution) access - neither observation nor modification
2. r(ead) access - observation with no modification
3. m(odify) access - modification with no observation
4. w(rite) access - both modification and observation.

The set of access attributes associated with each object is only one of the many characteristics required for the complete description of a system state.

The model's description of a system state consists of a four-tuple of the form: (current access set, access permission matrix, level function, hierarchy). The current access set is a collection of three-tuples indicating a subject an object, and the current access attribute of that subject toward an object. In other words, this set represents all the current accesses of all currently active subjects toward all objects maintained by the system. The access permission matrix is a two dimensional table in which all the rows of the table represent all system users (subjects) and the columns of the table represent all system objects. The

intersection of a particular row and column contains the explicit access attributes of a row's subject toward the intersecting column's object. In actual computer systems, the access matrix would be very sparse if it were implemented as a two-dimensional table. Thus, implementations of the access matrix table are more likely to be in the form of either a capability list or a access control list (10:257).

Perhaps the more interesting notion within a system state's description is the concept of level function, the foundation of security classifications within the model. The description of a level function begins by introducing what is meant by a security level, a classification and a category pair. In this context, classification is used in its usual sense, such as "a document has a certain classification" or "a user is cleared up to a certain level". A category is specialized compartmental information such as "NATO" or "Nuclear". Also introduced is the notion of security level dominance. A document's security level dominates a second document's security level if and only if the first document's classification is greater than or equal to the second document's classification and the security category of the second document is a subset of the first document's security category. This idea proves important when addressing the model's security related properties. The level function itself is defined to be the triple of the maximum security level of the subject, the maximum security level of the object, and the current security level of the

subject.

The final component of a system state, hierarchy, concerns the structure imposed on the objects. To take advantage of both the implicit control conventions and the wealth of logical data objects structured in a hierarchical manner, a parent-child relationship must be maintained in which only directed, rooted trees and isolated points are allowed (2:12).

The system characteristics which the model desires to maintain are collectively referred to as security. These characteristics are described implicitly by the ss-property, the *-property, and the ds-property. The ss-property, simple security, stipulates that if the three-tuple consisting of (subject, object, observe-attribute) is a current access, then the maximum security level of the subject dominates the maximum security level of the object. In other words, a subject cannot observe any object over which it does not dominate. The *-property states that if a subject has simultaneous observe access to one object and modify access to a second object, the security level of the second object dominates the first object's security level. This property prevents any subject from allowing any more highly classified information from flowing into a lower classified file. Together, these two properties constitute the nondiscretionary policy, as described in the previous section. The final security property, the ds-property allows an individual to extend to a second subject discre-

tionary access to a document, an access restricted only by the aforementioned nondiscretionary properties. A system state satisfies this property provided every current access is contained within the access permission matrix.

The second facet of the model, general mechanisms, introduces the inductive nature of security along with rules for the specification of system capabilities. The basic security theorem, which states that system security can be systematically guaranteed when each individual change to the current state does not in itself cause a breach of security, is the first general mechanism. In other words, the inductive nature of security is established by the basic security theorem in that it shows the preservation of security from one state to the next guarantees total system security (2:20). The second general mechanism provides a framework for isolating individual transitions within the model. This framework depends on a function, referred to as a rule, for specifying a decision and a next state for every state and request.

rule
(request, current state) ==========> (decision, next state)

Figure 3. Rules (2:21)

Each class of requests is analyzed separately in a rule which is designed to handle that particular class. Furthermore, total system response to the (request, current state)

pair is defined as the response of the rules written to handle that specific request.

The final mechanism, which Bell and LaPadula term a general development, centers on the relation of rule properties to the system properties. Basically, this development shows that the entire system specified by a set of rules satisfies all three security properties provided each rule does not introduce an exception to one of the properties. Together, these general mechanisms bound the scope of investigation to single state transitions.

One aspect of the model, not previously addressed, concerns the notion of trusted subjects. Trusted subjects are essentially hardware or software system components which are able to perform actions not otherwise permitted by the model's security properties. For instance, a system security manager must be given access to the access matrix to properly maintain access control. The software performing these actions on the security manager's behalf are considered trusted subjects. It seems that when tailoring the model to a specific environment, the model always allows for trusted subjects.

Grohn Model

The Grohn model represents a mathematical model of a generalized, protected data management system which uses the Bell and LaPadula model as its starting point (8:6). The Grohn model was built upon the essential components of the Bell and LaPadula model, changing or extending those com-

ponents where required. This section describes the major deviations from the Bell and LaPadula model and then demonstrates the consistency of the Grohn model with the relational approach to data management.

One of the primary differences between the two models concerns the set of access modes or attributes. Grohn states that the execute access is to be considered as a read (observation) access because the execute attribute is subject to the simple protection property, the tranquility principle, and the discretionary control principle. Similarly, the write access, described in the previous model, was shown to be a combination of an observation and a modification access. The observation access is defined to be the extraction or utilization of data from an object while the modification access involves modifying the object without any observation of it.

An important property for both of these models, although not explicitly mentioned in the earlier Mitre Corporation study, is the tranquility property. Essentially, this property states that the security levels of active objects will not change during normal operation. This property prevents both the over classification and the under classification of data. Furthermore, the tranquility property ensures that the clearance matching and the formal need-to-know requirements are observed.

The Grohn model introduces an additional property in order to maintain adequate controls over modification

access, mechanisms determined to be deficient in the Bell and LaPadula model. This new property, the integrity property, requires that the initial "soundness" of a system be maintained throughout its activities. Similar to the *-property, which was discussed as part of the Bell and LaPadula model, the integrity property requires that the integrity level of an observed object dominates that of a modified object, since otherwise, data subsequently modified with high level integrity may contain data of lower integrity. The requirement for direct and indirect modification control mandates that the integrity mechanisms be defined similarly to the aforementioned security mechanisms.

In order to simplify the procedures responsible for checking authorizations for every data access, the security and integrity mechanisms are combined, resulting in properties invoking a minimum of entities. To overcome the complexities such a combination entails, the properties are redefined in terms of the current levels of the subjects. As a result of these combinations, the complete set of mechanisms which cause data access in a system to comform to an access authorization policy are referred to as protection properties (8:26). All four previously defined properties, simple security, *-, tranquility, and discretionary, are similarly defined for the protection mechanisms. For instance, the *-property states that a subject may modify an object only if the protection level of the object dominates the current protection level of the subject.

In the Bell and LaPadula model, the basic access control mechanisms relied heavily on the directory system's hierarchical structure. In the this new model, the directory will serve the same purpose, yet accessing an object only involves one directory. Unlike the hierarchical structure, the new directory system permits searching for all dominated objects, a facility which is important for a general data management system (8:23). As a result of the elimination of the hierarchical directory structure, naming uniqueness problems arise because the Bell and LaPadula model used the path of the object to serve as a unique identifier. To overcome this problem and avoid violations of the *-property, the Grohn model includes the object's protection level as part of its identifier. Thus, similarly named objects but with different protection levels may coexist in the same directory. Additionally, an indication of the creating subject may also form part of an object's identifier, which would allow some validation of deletion operations to be performed. Although the directory system is transparent to the user, the modeling of its components ensures adequate consideration of this invaluable mechanism.

The final characteristic which distinguishes this model from the Bell and LaPadula's model concerns the nature of the object being dealt with in a data management system. In order to provide a database which is self descriptive and system independent, it is useful to distinguish the descriptive and value characteristics of the object. To make this

distinction, a three-tuple of the form (M,D,V) is used to represent every object. The M component depicts the column of the access matrix particular to the object and will contain all the accesses authorized for this object. The D component represents the descriptive aspect while the V represents the value aspect. Together, these three components provide a convenient way in which all active system objects, as well as the object's current status, are modelled.

The Grohn model, as just described, uses the relational approach to data management to demonstrate the consistency of the model for a number of reasons. Perhaps one of the most important reasons is the high degree of data independence provided by this particular approach. Furthermore, an equally important consideration is the simplistic and consistent approach used by the relational model. In the next few paragraphs, several data management mechanisms are depicted through the Grohn model. However, no introduction to the relational approach is presented as the reader is assumed to knowledgeable in this area.

An important issue regarding the consistency between the model and the relational approach to data management concerns the decision on how to translate a relational database into a set of protection levels. The alternative mapping options for choosing a data structure ranges between a data element, the most elemental object, up to the entire database itself. As a compromise between the simplest and

most the complex structures, this model chose the relation as its basic structure to which to assign protection levels. The justification given for this decision involved the fact that no restrictions on either tuple-oriented or domain-oriented activities occurs. Furthermore, the choice appears consistent with the notion of relations as units of data, since the relation will become a table of data classified at a single level. However, there are problems with this scheme such as the potential for data item overclassification and inconsistencies caused when low-level relations are deleted in spite of references made to them by high-level relations. In spite of the problems, the Sharp Corporation report states that using the relation as the model's basic object will not cause unauthorized disclosure (8:60).

The directory system, as described above, manages the unique identification of relations in terms of the relation's name and protection level. Besides a relation's name and protection level, the creating subject's identification and the relation's type (primary or derived) provide unique identification characteristics. Furthermore, each relation has an associated access matrix and a set of descriptor tuples for each domain. The domain descriptor tuples include such information as the domain's name, maximum length, and data type. Each relation also maintains a special tuple which contains status and size information.

The allowable requests in the system include directory, access matrix, descriptor or value, observations and modifi-

cations. In response to these requests, a number of different operations are performed such as create, observe, and delete. The complete set of operations implement the union, intersection, projection, selection, and join operators common to the relational approach to data management.

The final point regarding the consistency of the Grohn model to the relational approach to data management concerns multilevel activities and the protection level which should be assigned to views, which are derived relations composed of potentially different protection levels. Since the modification of a view only changes the view's definition and not any data, the Grohn model considers the view as an observation mechanism. Thus, the simple protection property becomes the criterion against which to make the protection level decision. Therefore, the protection level of the view must dominate all the protection levels of the base relations from which it is derived.

The Grohn model, as well as the Bell and LaPadula model, certainly provides for a rigorous description of a secure computer environment. However, how can one be absolutely sure that the model will actually provide the intended level of security? The answer lies within a number of recently developed techniques aimed at producing correct and reliable software.

Verifying Security

The protection of sensitive information is a difficult task, particularly when the data is maintained within a computer. Since one of the primary functions of a computer's operating system is to control and provide access to this sensitive information, the operating system often has the responsibility of enforcing the security policy. Thus, operating system security is the study of how to design software systems that successfully protect classified information and how to verify that a proposed design and its implementation actually provide the required protection. The requisite mechanisms for proving the consistency in the development process include specification and verification systems which explicitly support a hierarchical development of specifications. The hierarchical approach provides for the system's verification to be carried out at the different stages of development, from the abstract representation to the specific implementation of the system. In a typical system development, the primary representations included are the security model, the formal specification, and the implementation. Design verification shows the consistency between the formal specification and the security model while program verification shows the consistency between the formal specification and the implemented program (4:282).

To aid in the production of reliable software, automated specification and verification environments have been developed. These tools provide the methodology by which the

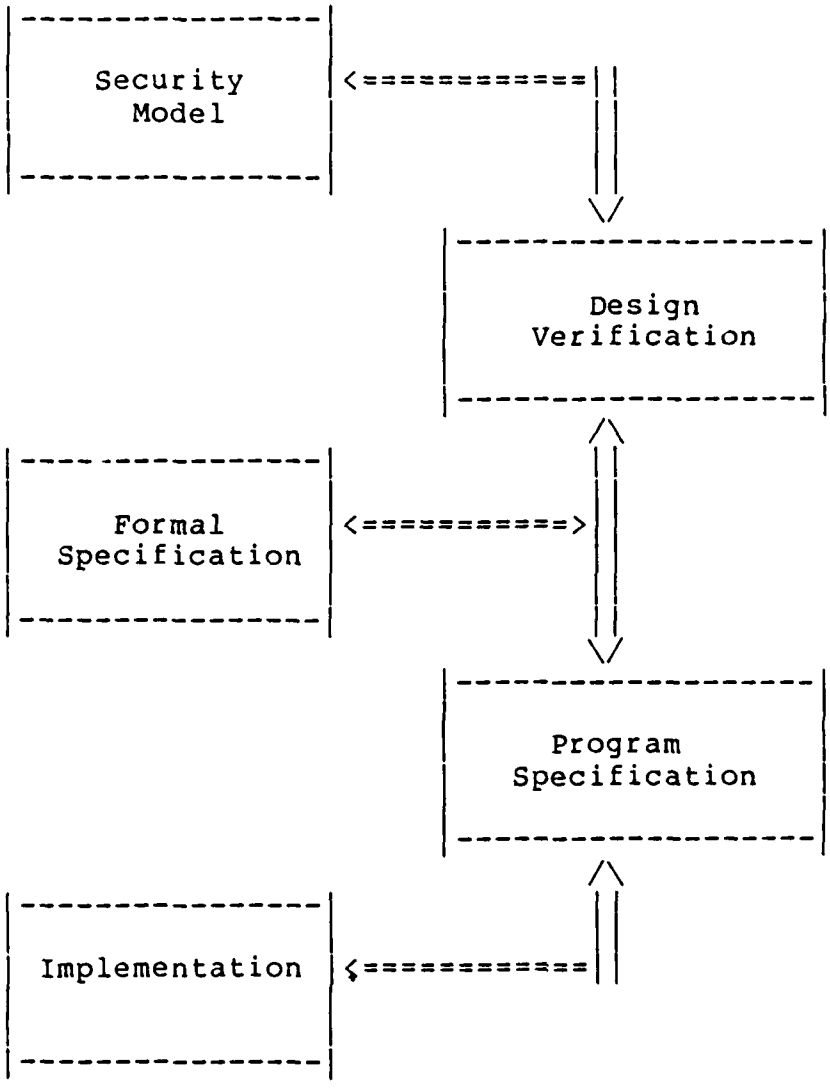


Figure 4. Development and Verification Hierarchy

aforementioned security properties are proven. The verification environments include the following components: specification language processor, verification condition generator, and theorem prover. Typically, each environment has its own unique specification language, which is derived from the language of mathematics, logic, and programming (4:28Ø). The specification language allows for the system

designers to state formally the functions provided by a given system of programs. The specification processor uses the formal specification as its input and generates logical expressions expressing security and consistency.

The verification condition generator is similar to a specification language processor in that it also produces as its output a set of formulas. However, the verification condition generator uses as its input a computer program and some assertions about that program. These assertions are inserted into the program at various points, indicating what program variable values are supposed to be at that particular point. The verification condition generator creates a verification condition for two successive assertions in the program, stating that the second assertion will be true if the first assertion was true and the code between the assertions was executed. A program is partially correct if all the verification conditions can be proven. However, total program correctness requires that the program cannot be caught in an infinite loop (4:281).

As the final component in the specification and verification environment, the theorem prover uses the logical expressions and formulas generated by the other two components as input. In one of the contemporary systems, Hierarchy Development Methodology (HDM) developed by the Stanford Research Institute, the theorem prover also uses as input any axioms or lemmas which might be useful to assist the proof. The final output of HDM's theorem prover is a

detailed English-like description of the proof process. The systems available today vary considerably in the degree of user interactiveness with the theorem prover. With the HDM, the user is totally out of the proof process. If the prover fails, no attempt at backtracking is made. The designer may input additional suggestions to the system to assist in the proving process before it is started over, obviously a time consuming operation. Other systems permit the user to interact with the theorem prover to assist in the more complex proofs.

In order to provide a better understanding of the specification and verification environment, an analysis of the previously mentioned Hierarchical Development Method is presented. The purpose of HDM is to produce highly reliable and verifiable software. In order to achieve this objective, the Stanford Research Institute has broken the system development process into several steps (see Figure 5). The entire development process is divided into two major stages consisting of the design specification, which is shown on the left side of Figure 5, and the implementation stage, which is shown on the right side. Data structure refinement, which maps data structures from higher to lower levels, is expressed in the mappings between the specification. Moreover, the algorithmic refinement is expressed within the implementation stage. Both the data structure and the algorithmic refinements express relationships which must be consistent between successive levels of the

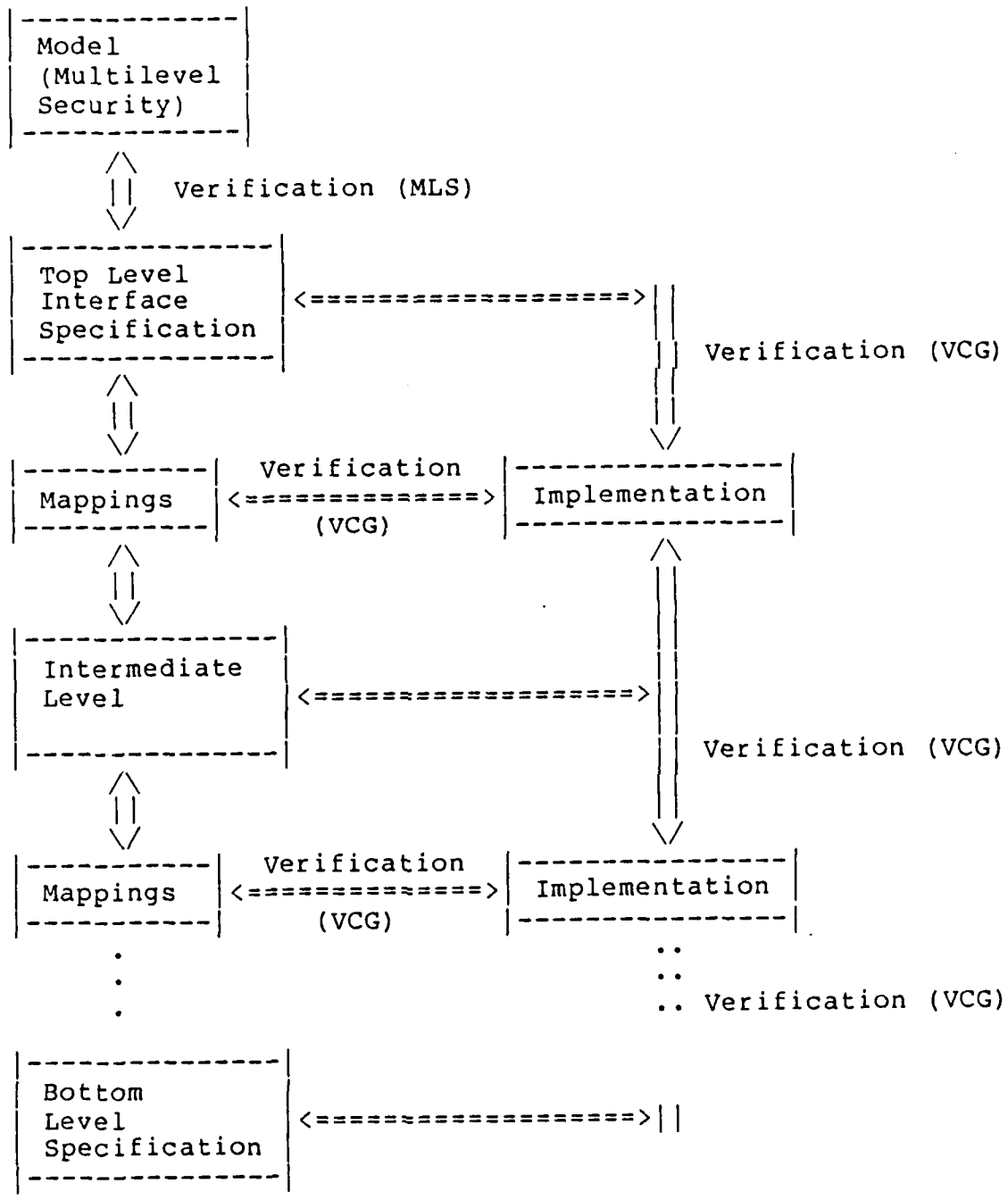


Figure 5. HDM System Development (4:287)

development process. Once the system's requirements have been described in terms of a model and a top level specification (TLS), the HDM uses a software tool called the multilevel security formula generator, along with the Boyer-Moore Theorem Prover, to perform an information flow analysis and prove correspondence between the multilevel security model and the TLS (4:286). The TLS should be the most complete description of the systems external behavior. To help achieve this level of detail, the TLS is written in SPECIAL, the specification and assertion language. The intermediate level specifications describe "abstract machines" where each successively lower level contains more primitive functions of software and hardware. Between adjacent levels of specifications, mappings written in SPECIAL define how data structures are translated from a higher level into the lower level representations. However, implementation is necessary before lower level operations can be shown to provide the primitives to carry out the upper level requirements.

The second stage of the HDM, implementation, involves writing source code in some high-order language for which there is a compiler available for the target machine. In order to accomplish program verification, a verification condition generator (VCG) must be available for the chosen programming language. In the HDM, the implementation is written in Modula and is then translated into the Common Internal Form language used by the VCG. As with the multi-

level security formula generator, the Boyer-Moore Theorem Prover is used to prove the verification conditions generated by the VCG. The implementation consists of several procedures defining how each of the operations performed in a given "abstract machine" are implemented in terms of the data structures and operations in the next lower level "abstract machine". Using these two specification levels and the mapping between them, the Common Internal Form representation of these procedures is verified correct.

The specification and verification methodologies seem quite complex and require considerable effort and training to completely understand and properly use them. However, these specialized systems truly enhance the secure computer system designer's capabilities to prove the correctness and reliability of "trusted" software.

A final topic, not previously addressed in this section, concerns the notion of secure computer system evaluation and certification. In this regard, the Department of Defense's Computer Security Center issued criteria against which to evaluate the specification and verification procedures used to develop trusted computer systems. The criteria, as established by the Computer Security Center, serves three purposes. First, the criteria provide a standard with which to evaluate the trustworthiness of a computer system for the processing of sensitive information. Additionally, the criteria provide guidance to manufacturers as to what

security features should be built into future systems to satisfy the trust requirements for classified applications. The final reason for establishing these criteria is to provide a basis for specifying security requirements in acquisition specifications (7:2).

The criteria establish four divisions into which a particular computer system may be classified. The first division, D, with only one class, provides minimal protection. Division C, with the two classes of discretionary protection and controlled access, provides need-to-know protection and through the inclusion of audit capabilities, provides for a subject's accountability for the actions they initiate. The third division, B, contains the three classes of labeled security, structured domains, and security domains. This division uses the notion of a trusted computer base (TCB), which is defined as the totality of protection mechanisms within a computer system responsible for enforcing a security policy (7:114). Computer systems within division B must use security labels with all major data structures. The system designer must provide the evaluator not only the formal security model on which the TCB is based but also the TCB specification. The last requirement for this division is that evidence must be demonstrated that the reference monitor concept has been implemented. Finally, division A, which at the present time has only the verified design class, is characterized by the use of formal security verification methods. Extensive

documentation is required for this division to demonstrate that the TCB meets the security requirements in all phases of design, development, and implementation. As envisioned by the Computer Security Center, the use of formal verification techniques will be extended to the source level as better techniques become available. Consideration will also be given to the correctness of the tools, such as compilers and assemblers, used in the TCB development.

Computer evaluation and certification, as described above, is an extremely challenging task which is taken very seriously by the Department of Defense. Each successive division provides tougher and tougher requirements to meet in order to "prove" the trustworthiness of a perspective system. As the tools for demonstrating the system's security mechanisms are perfected, one would expect not only tougher criteria, but also a more thorough attempt at demonstrating the security of the entire development process.

Conclusions

The Department of Defense seems to be confident that the design of secure computer systems using the kernelized approach is the best short-term solution to this complex problem. This opinion is confirmed by many notable computer scientists, including Stanley Ames, when he states that "the security kernel design approach is the most promising methodology currently available that can provide both the internal security and the functional capabilities that many of today's computer systems need (1:22)." Based on both the

number of currently existing systems and those in the development process which use the kernel approach, this author would agree with their optimism. However, there are a number of unresolved problems.

One of the most significant problems associated with the kernel design is the confinement problem, which is described as the prevention of a program from leaking sensitive information. The kernelized approach does not address "indirect" information leakage channels, which work through the operating system, by means of response codes and other intended paths (4:284). However, a procedure to detect "indirect" information flow was implemented at the Stanford Research Institute (4:285). A second type of security model, proposed by Denning, specifically addresses the confinement problem and can be certified capable of verifying partial or total confinement of a procedure (6:511).

A second important problem concerns the current state of the specification and verification methodologies. Most of these systems are considered experimental and should not be regarded as a final product (4:280). Again, Mr. Ames even supports this view in his article when he states that solutions for the program correctness problem are still a long way off, particularly for large programs such as an operating system. However, the Ames's article also stated that "even if there was no intent to complete a full mathematical proof, we still have the rigorous review,

documentation, and kernel development guidelines that most verification methodologies enforce. These alone will ensure a more secure and reliable system" (1:21). This opinion on Mr. Ames' part raises another, perhaps unanswerable, question, "How secure is secure?"

A final comment of the kernelized approach relates the desired performance objectives with the results observed to date. Naturally, one of the primary objectives of the kernel approach was for the enhanced operating system to minimally degrade system performance relative to its non-kernelized counterpart. Unfortunately, performance has been a serious problem for early attempts at kernelized secure operating systems. In fact, some of these early systems have provided only 10 to 25 percent the performance of similarly configured nonsecure systems. Recent implementations have been more successful at producing systems with "adequate" performance (11:87). Again, the question could be asked, "How minimal an impact on system performance is truly minimal?"

The bottom line to this entire discussion is that providing a verifiable, secure operating system is a complex problem which is not completely resolvable with the tools which are available today. The use of the kernelized approach is, at best, a partial solution to one of the most challenging problems in the field of computer science.

IV. Network Design

Introduction

This chapter examines the alternative design strategies for the development of the secure local area network described in Chapter II. Each approach is analyzed with respect to the specific security mechanisms it requires for implementation, particularly those mechanisms which are attainable using currently available technology. Additionally, four different local area network topologies are investigated with particular emphasis on their security related mechanisms. However, before addressing the specific design alternatives, requisite background material is presented.

Background

The two major issues which must be addressed when considering various secure local area network (LAN) architectures are user/subscriber separation and data protection. User separation refers to the ability of the LAN to provide segregated subscriber communities. In other words, some network component must ensure that a user cleared for level zero information receives only that information classified up to level zero. The enforcement of such a user/subscriber separation policy provides the necessary protection against the accidental disclosure of sensitive information to the authorized but uncleared LAN user. The second major issue, data protection, refers to

the ability of the network to provide protection against malicious attempts to access or modify network resources. Currently, there are two approaches to providing such protection, protected wireline distribution systems (PWDS) and data encryption. A PWDS is a telecommunications system to which physical safeguards have been applied to permit safe electrical transmission of unencrypted data while encryption renders all communications unintelligible except to the intended recipients of that data (16:B-3). However, recognizing the importance of both of these issues, the efforts of this thesis are concentrated on user separation due to the nature of the specified operating environment.

Since a comprehensive architecture must be defined in terms of a specific installation, an extensive examination of the vulnerabilities peculiar to that site provide the LAN designer an awareness of the tradeoffs to be considered during the development process. Vulnerability, in this context, is defined to be a design or implementation flaw which may cause any component of a LAN to operate in a manner that differs from its specification. This potential flaw is realized through either the intentional or unintentional action of some LAN process or subscriber. Moreover, such conditions exist within all systems, regardless of architecture, access method, or physical protections. Various design methodologies afford differing levels of protection which are equal to the design characteristics minus any inefficiencies in the

implementation of the security mechanisms (16:2-1).

Generally, the actions which exploit these apparently inherent LAN weaknesses are classified under one or more of the following categories: disclosure, modification, and denial of service. Perhaps the most important category of the three, as far as this study is concerned, is disclosure. As defined in a recent System Development Corporation report, disclosure is "the resultant product of events in which any person, group, or entity views, records, or gains unauthorized access to system resources (16:2-2)." Disclosure is accomplished either through accidental or malicious means. For example, suppose a user was logged on to computer B{M} and requested level two information from computer A{L}. However, due to some unique combination of events, the user is actually presented with level zero information from computer C{H}, an unauthorized yet accidental disclosure.

The second category of LAN weaknesses is the unauthorized modification of system resources. For this study, as was mentioned in an earlier chapter, the environment is assumed to be free of both accidental and malicious instances of this LAN weakness. Denial of service is the final LAN weakness category. Although this category generally includes any act which precludes system users/subscribers from authorized resource utilization, it also encompasses impairment of LAN resources. A question which now must be considered during the design process is,

does a LAN security component which noticeably impairs system response time constitute a denial of service weakness in and of itself? This author believes any unreasonable delays do constitute such a weakness, although a very subtle weakness at best. Furthermore, it is hoped that an implementation of the proposed design will not impose any of the weaknesses just discussed; although, disclosure is the specific weakness this study attempts to eliminate.

The network security problem, like all security problems, exists because authorized or unauthorized network users could potentially "misuse" LAN resources if given an opportunity. The nature of these "hostile elements" and the resources to be protected, leads to the development of an appropriate security policy. A security policy, which is the formal expression of a set of strategies for countering all recognized threats, when properly implemented, leads to a prescribed level of protection specific to each LAN being designed. As was pointed out earlier, this protection is never absolute and in most cases doesn't extend beyond some predefined set of threats. However, the security policy and requirements issues related to how these threats are to be countered, establish the high-level constraints for this study. Furthermore, these policy issues define what functions the security mechanisms provide. Subsequent sections of this chapter discuss specific security mechanisms; however, before proceeding, general LAN protection issues are discussed.

The primary protection issues when developing a LAN include identification/authentication, access request/ authorization, access control, security monitoring, and security assurance (12:11-38). The first of these issues, identification/authentication, involves procedures for determining and verifying the identity of the person or device requesting access to the network resources. Generally, the identification of the requestor is done through an administratively assigned login identifier such as the initial of the user's first name prefixed to the letters of his last name, up to some maximum length. The authentication issue involves the verification of the requestor's identifier, typically through the use some user defined password. Resolution of the following authentication issues must be accomplished before the LAN design is completed: means of authentication of persons and devices, process authentication, and distributed versus centralized authentication verification. Access request/authorization issues concern the user/subscriber capabilities to read from, write to, or execute on a specific LAN resource once his identification has been determined and verified. As was discussed in Chapter III, the information which defines the requestor's rights to protected system resources is maintained in a "capabilities" list. This list consists of a user's profiles which identify the objects this particular user has access to and the explicit actions the user may take in regards to these objects. A number of different

access request/authorization techniques are described in the literature, with the most common technique being the use of a reference monitor. Briefly, the reference monitor is a software mechanism which monitors access requests to system resources by requiring its invocation every time an access attempt is made. The software mechanisms which constitute the reference monitor are considered trusted components as discussed in Chapter III and are the arbitrators in all access decisions.

The third protection issue involves access control as related to the establishment of additional connections within the network and the necessary security mechanisms to detect unauthorized connection attempts. The designs of network security mechanisms gain a substantial part of their strength by their ability to control the creation of communication paths between the requestor and the resource. The establishment of network connections in a controlled manner concerns the creation of a logical or physical path through the network, the use of appropriate control disciplines including encryption mechanisms, as well as the flow of identification and authorization data between requesting and resource computers. The key is to present "multiple barriers through which a potential penetrator must pass (12:25)." Security monitoring, the fourth issue concerning network protection, provides the wherewithal to detect and resolve security problems should one of the other mechanisms fail to detect improper network use. Basically,

security monitoring is the provision of an audit trail which records all LAN activities. Again, the network designers must consider the tradeoffs between local and global security monitoring. One global approach has been the use of a network security center which supports the needs of correlating and interpreting audit information. The final network protection issue involves security assurance. Fundamentally, assurance issues concern both hardware and software reliability and the degree of confidence LAN designers can have in these network components. Even though this issue was discussed in great length in Chapter III, let it be said again that the software assurance issues are the critical part of this study.

This background material has provided a brief look at the many issues to be addressed in the design of any secure local area network. All of these issues are considered before a choice is made of a particular design but only a few of them are actually documented in the next section dealing with the specific LAN architecture.

Secure LAN Architectures

Over the past few years, many secure local area network architectures have been proposed by both government and industry. These designs, being by necessity very general in nature, provide a good foundation upon which to tailor the specific network being developed in this study. The architectures studied in this section include a multiple channel network, trusted interface units, a security

controller, and a distributed secure system.

Multiple Channel Network. A multiple channel LAN provides for user separation by dividing up the frequency spectrum among the various logical channels. This frequency division multiplexing provides a frequency band dedicated to a specific data classification level for the exclusive use by each particular user/subscriber group. This design, using a single coaxial cable, is incorporated into a broadband network using either fixed frequency or frequency agile modems (16:2-12). Figure 6 depicts the various computers as described in Chapter II with both the hosts and the cable being installed in physically secure locations. As depicted in this figure, each host is connected to the LAN subnet through a bus interface unit (BIU). Each BIU is responsible for the appropriate modulation/demodulation activities.

The basic model for this design uses a fixed frequency modem with each different data classification level using a different fixed frequency. However, for the application being considered in this study, this is inappropriate because of the inability of the hosts to communicate with each other due to the potentially different data classification levels each would maintain. However, a variation of this basic model overcomes this difficiency. The extended multiple channel network architecture includes frequency agile modems; thereby providing the capability to access a number of logical channels. Using this

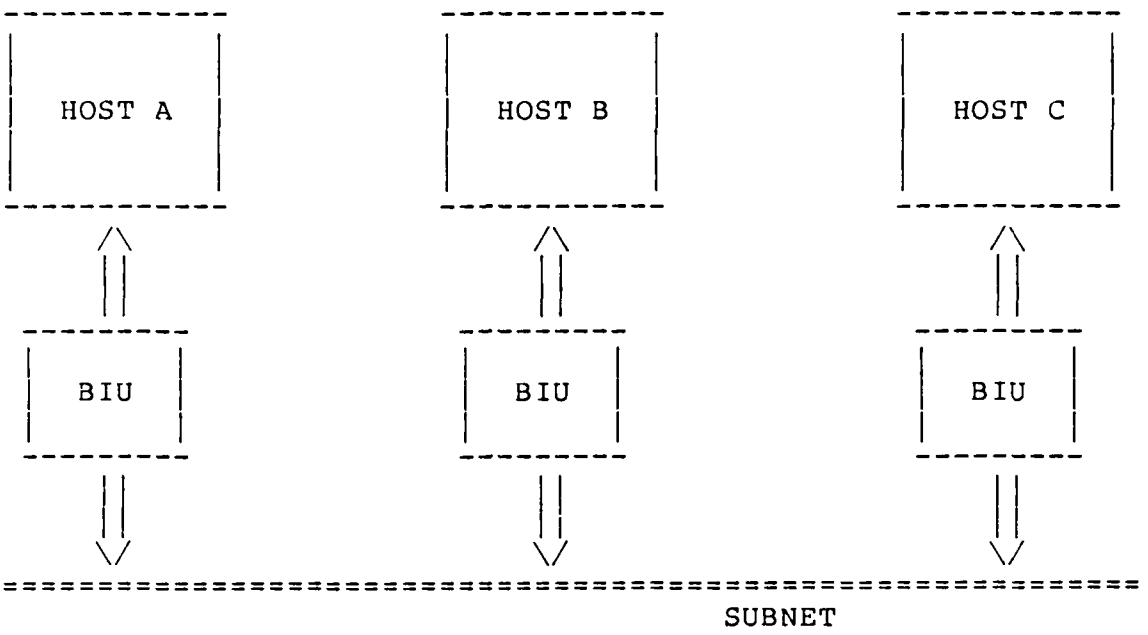


Figure 6. Multiple Channel Network

variation would allow computer C{H} to communicate with both computers A{L} and B{M} while computer B would be able to communicate with computer A{L}. In other words, computer A would have a fixed frequency modem while computers B{M} and C{H} would have frequency agile modems. Computer B{M}'s modem would be able to access computer A's channel in a non-simultaneous mode using either local or remote switching control. Similarly for computer C{H} except it would also be able to access computer B{M}'s logical channel. Support of computers B{M} and C{M}, which require access to multiple channels, is provided through the use of a bridge which links a single host channel to a number of segregated channels, thereby providing for simultaneous monitoring of

all access channels accessible by each respective host (16:2-16). In the case of a host which is multiple level secure, each port is dedicated to a single security level (channel) and a separate modem will be supporting that port. The question now arises as to which component of the architecture provides the requisite security mechanisms. Besides the logical separation of channels provided by frequency division multiplexing, the entire burden is the responsibility of each host's operating system. As reported in the System Development Corporation study, each host is "assumed to provide protection against unauthorized disclosure of restricted information (16:2-14)."

Trusted Interface Units. In this architecture, special interface units enforce the data security policy. These special units, referred to as "trusted" interface units (TIUs), perform security label checking on both data placed upon and received from the LAN subnet. In those environments where tapping into the subnet is a potential problem, physical protection of the subnet is required to ensure the TIUs only provide contact between authorized network subscribers. As was described in Chapter III, a "trusted" interface unit implies a device with sufficient hardware and software integrity to allow for its simultaneous processing of multiple levels of sensitive information. Generally, a trusted interface unit allows an untrusted subscriber/process to access the LAN. To enforce the policy that only equal security level processes

communicate, the TIU checks the data received from the LAN to confirm the equality of the received data's security level and the security level of the TIU's associated host. The TIU also ensures that all data placed on the LAN medium by the untrusted process is correctly labeled. In those cases where classified processes in different secure networks want to communicate, bridges connecting the two networks must enforce similar security policies.

As the means of gaining access to the LAN, the TIU is responsible for at least the functions performed by the physical and data link layers of the International Standards Organization's Reference Model of Open Interconnection (16:2-27). In other words, the TIU is concerned with transmitting raw bits over the subnet medium and transforming those transmission into a line which appears error free to the higher layers in the ISO model. The TIU is also responsible for the processing of security labels which are consistent with attached, untrusted processes as was described above. Figure 7 illustrates the logical construction of the multifunction TIU.

Besides the assurances that the physical channel performs correctly and that the data link layer never modifies the security labels, the TIU must, most importantly, be trusted to enforce the security policy of internode communication. For example, if the TIU handles a single security level, this security level can be permanently integrated into the

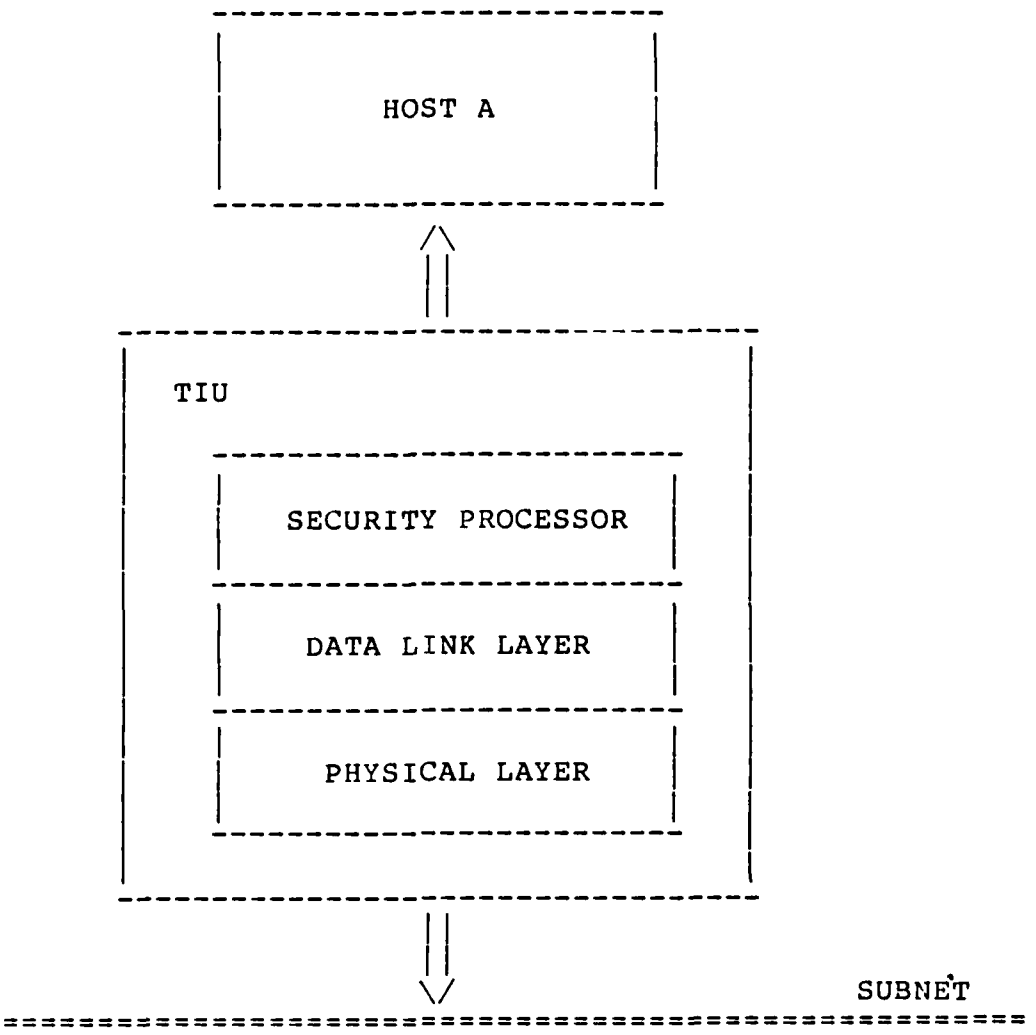


Figure 7. Trusted Interface Unit Logical Construction

TIU's hardware or firmware. However, if the TIU handles multiple security levels, each time there is a change of classification, all sensitive residue must be eliminated from the device before communication is allowed at the new security level. Existing security policy dictates whether this sanitization is user controlled or performed by an access controller.

As was mentioned briefly above, bridges are required for

internet communication. For the purposes of this study, a bridge would be used to connect the LAN to the external communications network (see Chapter II for a description of this external network). Functionally, the bridge resembles back-to-back TIUs; that is, one TIU on the LAN side of the bridge to ensure properly labeled messages are sent to and received from the external network and a TIU on the external network side of the bridge to perform similar functions as those of its counterpart on the LAN. The construction of the bridge in such a manner maintains each network's internal integrity more so than if the bridge's functions were performed by an individual TIU. Additionally, the bridge may be functionally more complex due to the fact that the internet routing algorithm may differ from the routing algorithm used internally to the LAN. Similar to the TIU, the bridge must be trusted to properly perform its security filtering and to manage the integrity of multiple data streams.

Security Controller. The design for this architecture requires the three host computers (A, B, and C) to be under the control of a security controller (see Figure 8). The security controller (SC) is a host-level computer which provides the majority of the network security mechanisms. The specific functions performed by the SC, an independent third party mechanism, include validation of users/subscribers, providing authorization for access to LAN resources, providing secure communications paths, and

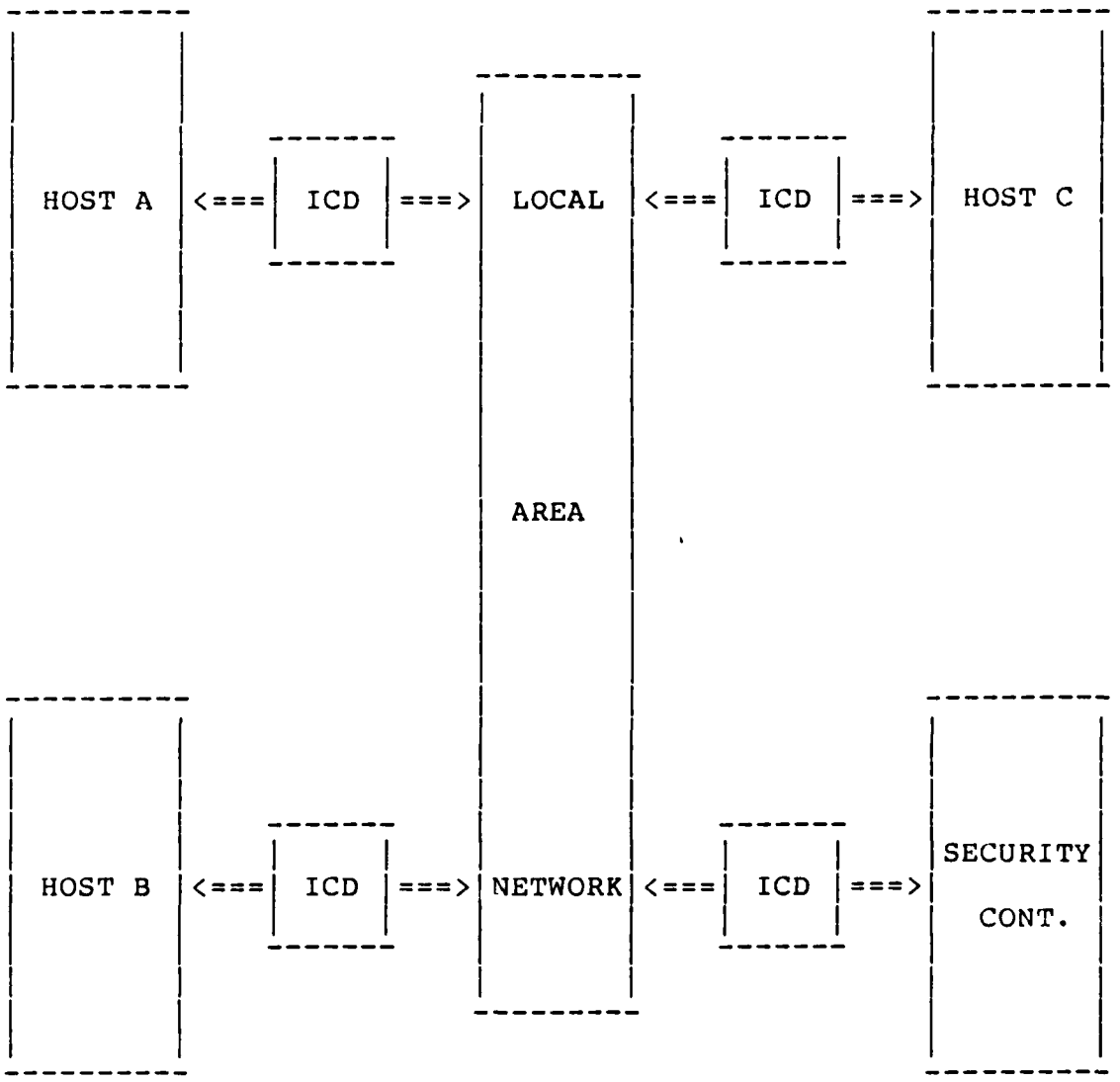


Figure 8. Security Controller Architecture

security monitoring (12:42).

The security controller performs the necessary authentication and authorization functions which were performed by the individual hosts in the previous two architectures. Thus, the SC must check the identification for all users/subscribers of the LAN. Obviously, due to this large responsibility, identification and authentication mechanisms would be developed as a separate and self-contained module within the SC. Once the SC has authenticated the user's validity, the SC must send this information to the LAN node, referred to as the serving site, maintaining the requested information. However, this necessitates the SC being able to identify itself to the server site. The two approaches to this problem are first, by providing each network node a unique password only known by that specific site and the SC, and secondly, by the use of an implicit scheme which takes advantage of the security controller's responsibility for remotely keying each of the cryptographic devices depicted in Figure 8. (More will be said about the details of this second scheme later in this section.)

Once the user/subscriber has been identified, the SC must determine if the user's request to LAN resources is authorized. As was discussed earlier in this chapter, this is accomplished through a table look-up. Questions such as what information should be in the table, table organization, and table updates are not addressed in this study. However, the answers to these questions concern not only how

efficient the SC performs its functions but also how complex its software resources become which in turn impacts the software verification actions required to assure the LAN designer that the SC can properly perform these functions.

One of the most important functions performed by the SC is in establishing the connection between the requesting and server sites. Since the SC performs a reference monitor function necessitating that the SC must always be invoked, the SC would set up a temporary set of working keys to protect any subsequent requestor-to-SC communication. The keys would be used in the encryption devices located logically between the SC and the subnet and between the requestor and the subnet. The next activity performed by the SC, once the request has been authenticated and authorized, is to set up a protected working connection between the requestor and the serving site. This is accomplished by providing working keys to the cryptographic devices at both requestor and server nodes via special control messages. Again, as there are several techniques for the communication of these special messages, the specific technique chosen is not addressed in this study. Any number of anomalies could occur during this connection process; therefore, all actions throughout the network are logged by the security monitoring software within the SC.

An equally important function performed by the SC is the protection of the working connection from misuse. Two potential problems originating from within either the

requesting or serving hosts are attempts to multiplex multi-user data over a single user connection or improper labeling of sensitive information. Some controls over these accidental misusages are provided by the network since checks can be made on the validity of addressing and classification fields. However, the primary responsibility for the resolution of these problems remains within each host. In other words, each host's operating system, as was the case in the two previous architectures, must be certified that it can provide the necessary assurances that these problems cannot occur. Once the communication between the requestor and server sites is completed, the SC is notified and the connection is broken.

The cryptographic devices play an important role in this architecture. In addition to the simple point-to-point protection provided by these devices, the cryptographic units provide protection against unauthorized connections between network nodes. Furthermore, these devices must be capable of being remotely keyed by only the security controller, as well as the provision for protection against the piggy backing of unauthorized users once a requestor-to-server dialogue has been completed. In order to provide these protection mechanisms, the cryptographic devices require a certain amount of logic. The control logic built into these devices depends on a number of factors including whether the cryptographic devices are multiplexed or dedicated and whether there is is a master-to-slave

relationship between cryptographic devices (12:104). These intelligent cryptographic devices (ICDs), as they are referred to in the National Bureau of Standards report, forward authentication and authorization information to the SC. In those instances where hosts dynamically change their security level, the ICD must be provided with two or more identifiers and authenticators so that it can serve in multiple roles. Obviously, this adds to the complexity of the device. Additional complexities are due to the required communications when a requestor-to-server dialogue terminates. Once the ICD is notified by its associated host that the communication has been terminated, the ICD must notify the SC so it can complete its audit record. This notification is accomplished by requiring the ICD to first establish an enciphered connection similar to the request initialization message described earlier. Then a preformated message would be sent indicating the termination of the dialog. After the notification to the SC, the ICD must be reset to receive its new private key.

The gateway to the external network in this architecture is handled in a very straightforward manner. The gateway is considered as an intermediary host which happens to reside in two networks. Typically, one of these networks is a local area network while the other is the global network of gateways that connects various LANs together. For the purposes of this study, the gateway would directly connect the LAN to the external communications network without the

benefit of the global network.

Distributed Secure System. The distributed secure system architecture (DSS) approaches the network security problem as a combination of the security mechanisms proposed in the three previously discussed architectures (13:55-67). However, the primary distinction of the DSS approach from the other architectures is that each host within the LAN is untrustworthy. Therefore, the security of the overall LAN must not depend on assumptions concerning the hosts' behavior. An important implication of the untrustworthiness of each host is that the unit of protection concerning host behavior must be the host itself. In other words, each host is dedicated to a single security partition. Before introducing the specific architecture, a general approach to the LAN security problem, as relevant to the DSS, is presented.

The DSS approach to the design of a secure local area network involves four separation techniques. The first of these techniques, physical separation, is achieved by allocating physically different resources to each security partition. To achieve security in a distributed environment such as the one described for this study, reference monitors must provide control of communications between the LAN components. Physical separation is also provided between each host and the security processors which host the reference monitors. Temporal separation, the second separation technique, allows the untrusted hosts to be used

for tasks in different security partitions by separating these tasks in time. The system state is reinitialized or sanitized between tasks belonging to different partitions. In order for each of the security processors to support a number of different separation and reference monitor functions, a separation kernel (13:57) is used to provide a logical separation of these functions. Simple security kernels whose only function is to provide separation can be relatively small and uncomplicated with verification simpler and potentially more complete than for general purpose security kernels (14:144-155). The final separation technique, cryptographic separation, involves encryption and uses checksum techniques to separate different uses of the shared communications and storage media. These four separation techniques, physical, logical, temporal, and cryptographical, provide the basis for the distributed secure system. The primary aspect of this approach is the enforcement of the prescribed security policy on the flow of communications between components of the LAN.

As was mentioned above, the security processor for each host is physically separated from the hosts themselves. These processors, referred to as trustworthy network interface units (TNIUs), mediate all communication between its associated host and the LAN subnet (see Figure 9). The basic model for the DSS architecture places the restriction on the LAN such that the TNIUs could only communicate with similarly classified TNIUs. However, as will be described

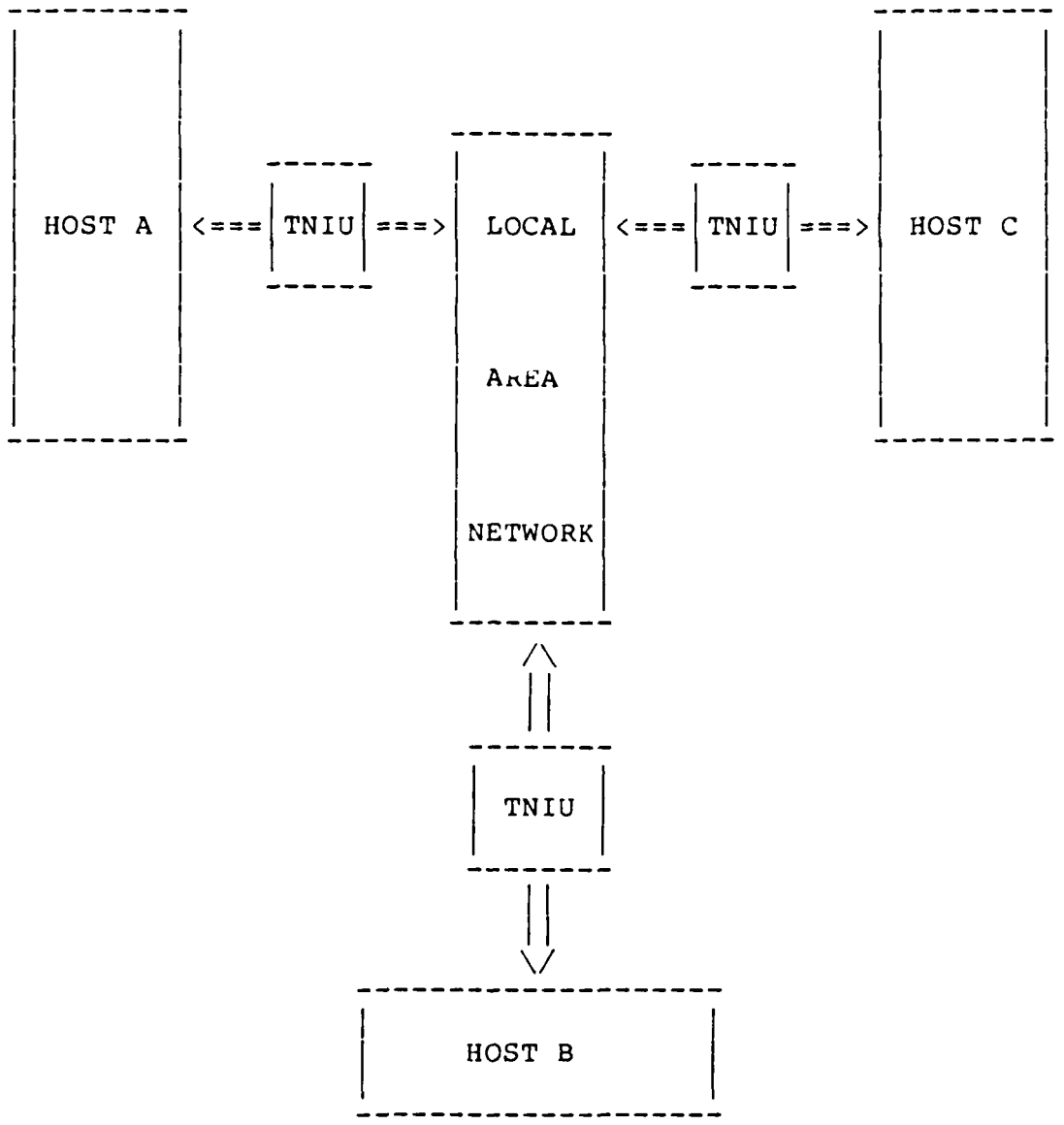


Figure 9. Distributed Secure System Incomplete

later, a variation of this basic model provides true multilevel security. To provide legitimate host-to-host communication channels and to protect the LAN from being subverted or tapped, even though this is not a problem in

the environment described for this study, the TNIUs encrypt all communications sent over the LAN. Since hosts are untrustworthy and may attempt to accidentally thwart the cryptographic protection, the encryption must be managed carefully.

Although encryption enforces separation between authorized and unauthorized LAN users, it does nothing to prevent an incorrect delivery because the LAN hardware accidentally misinterprets message destination fields, which are in clear text. In order to prevent the accidental disclosure of sensitive information should such an occurrence happen, the DSS uses two techniques which securely separate communication channels belonging to different security partitions. The first technique uses a checksum to assure the integrity of each message unit. Before encrypting each message, the TNIU calculates the message's checksum and then encrypts the checksum and the message as a single unit. The TNIU receiving the message must first decrypt the message and recompute its checksum. Only after the two checksums have matched will the TNIU accept the message for further processing. Thus the message integrity is guaranteed which implies the TNIU can trust the security partition identifier associated with each message the TNIU receives and reject those bearing a noncompatible identifier. The second technique to prevent accidental disclosures uses a different encryption key for each partition. Each TNIU is provided an encryption key unique

to its security partition which prevents any dissimilar partitioned TNIUs from communicating with one another. If a TNIU accidentally receives an unauthorized message, the different encryption keys will render the message unintelligible. Together, the checksum along with the differing encryption keys, prevent an accidental disclosure from ever occuring as a result of a misinterpreted message destination field. One further enhancement of the TNIU includes the provision for sequence numbers and timestamps to prevent spoofing. However, as the nature of this study's environment is nonmalicious, the last two enhancements are not used, thereby simplifying the TNIUs software.

Due to the interposition of the TNIU between the host and the LAN subnet, assigning functions to the layers in the protocol hierarchy is quite complex, especially when encryption is performed. For this reason, the TNIUs perform all protocol functions except those associated with the highest layer (13:61), thus relieving the hosts of the low-level network load and improving overall host performance. The top level protocol provides a remote procedure call (RPC) service which involves a reliable datagram facility serving as the host-TNIU interface. Essentially, all requests for services result in procedure calls. If the required object is remote, the top layer protocol intercepts the procedure call and substitutes a remote procedure call in its place. Once the remote call has been substituted, the host sends the procedure call to the TNIU. Upon receipt

of the RPC, the TNIU must validate the call because only specific RPC's are allowed in the LAN being developed in this study. For example, all database queries are valid, perhaps not fulfilled for security reasons, but at least the request is forwarded by the TNIU. Requests to access other remote resources, excluding the LAN activitites logs and the external communications logs, are disallowed by the TNIU. In other words, only predefined requests for remote resources are accepted by the TNIU. This limitation is imposed because in those instances where similarly partitioned hosts are communicating with one another, the access control lists are maintained by the host's untrusted operating system. Therefore, in order to prevent an unauthorized disclosure because of the absence of a need-to-know for a particular resource, such remote requests are prohibited.

In order to simplify the TNIU's software, the datagram service only provides the facility for the transmission of data on a "best effort" basis (15:60). This service does not include either the provision for datagram flow control or end-to-end acknowledgements because of the assumption made in regards to the high reliability of the underlying hardware. If any of the features of the transport layer are desired, such features must be implemented specifically using the datagram service. Obviously, such enhancements improve the efficiency of the datagram service but also increase the complexity of the required software.

As was just described, the TNIU performs numerous functions; however, as argued in Mr. Rushby's article, these devices are similar to cryptographic front ends already in use in long-haul networks. The addition of a single chip implementation of the Data Encryption Standard algorithm along with a separation kernal including appropriate memory management chips to the already existing hardware make the construction and verification of the TNIUs a realistic and attainable project. Furthermore, since disks are not required for a TNIU, all software could be maintained in read-only memories.

Thus far, the DSS design has restricted the flow of information between similarly partitioned hosts. To provide for the secure flow of information across partition boundaries requires a second type of trusted intermediary. The services provided by this intermediary include multilevel secure storage and retrieval of files. In the DSS, the only objects allowed to cross security boundaries are files; which, for the purposes of this study, would be no larger than a file since only requests for and responses to database queries are being passed between the LAN nodes. The trusted intermediary, referred to as the secure file store, has the ability to communicate with hosts of all security partitions. For example, the secure file store would receive computer C{H}'s query on the weather portion of the database which is maintained by computer A{L}. The query, which is made through a remote procedure call, a

mechanism to be described later in this section, is forwarded to computer A{L}, once all the appropriate checks have been by the secure file store. Computer A'{L}s response to this query follows the same step, just in reverse. Subsequent to the receipt of computer A{L}'s response, the secure file store will forward this file to computer C{H}.

In order to make the construction of the secure file store as simple as possible, it is physically housed in two separate components. The first component, which is trusted and is referred to as the secure file manager (SFM), is responsible for enforcing the security policy; whereas, the second component, which is untrusted and is referred to as the isolated file store (IFS), provides the SFM storage facilities. The SFM is a reference monitor responsible for monitoring communications which attempt to cross security partitions. Basically, the SFM is nothing more than an enhanced TNIU, whose internal structure is slightly more complex because of the multiple encryption keys required for the SFM to communicate across security partitions. If communications with different partitions occurs simultaneously, clear text belonging to logically different channels should be managed by separate virtual machines. Furthermore, temporal separation must be provided for different uses of the SFM Date Encryption Standard chip (13:63).

The specific tasks the SFM must perform are monitoring

access references and guaranteeing file integrity. To accomplish these tasks, the SFM is logically decomposed into a file access reference monitor (FARM) and a file integrity guarantor (FIG). The FIG performs its tasks through the use of checksum techniques similar to those used for LAN messages by TNIUs. Additional FIG responsibilities include time stamping all files sent to and received from the isolated file store. The FIG's tasks, along with those assumed by the FARM, are not significant complications and would place only a small additional burden on existing TNIU construction. As was mentioned previously, the enhanced TNIU housing the SFM would include a separation kernel in which the FARM software would be maintained. Therefore, all SFM functions can be integrated into existing TNIU designs which would require little additional development and verification costs over and above what is currently expended on the unenhanced TNIU.

The IFS, as was mentioned earlier, is an untrusted host-level machine. In order to maintain the integrity of files sent to and retrieved from the isolated file store, the secure file manager calculates a checksum and encrypts the checksum and the file as a single unit before sending it to the IFS. The FIG checksum mechanism allows files to be read from or written to the IFS only in their entirety. As an additional measure to ensure that the file requested by the SFM is truly the one it receives, each file is time stamped and encrypted along with the checksum and file (13:64). The

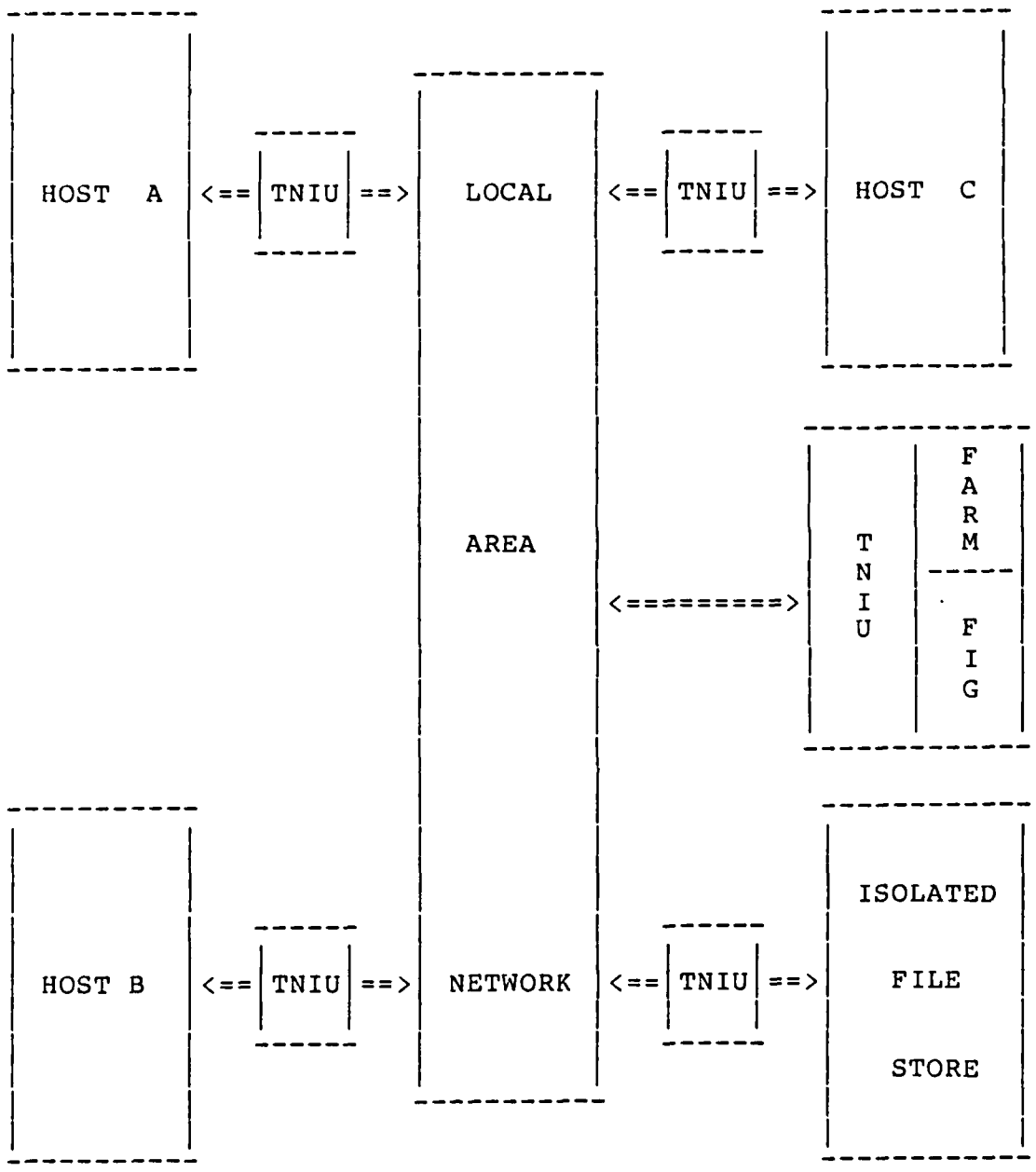
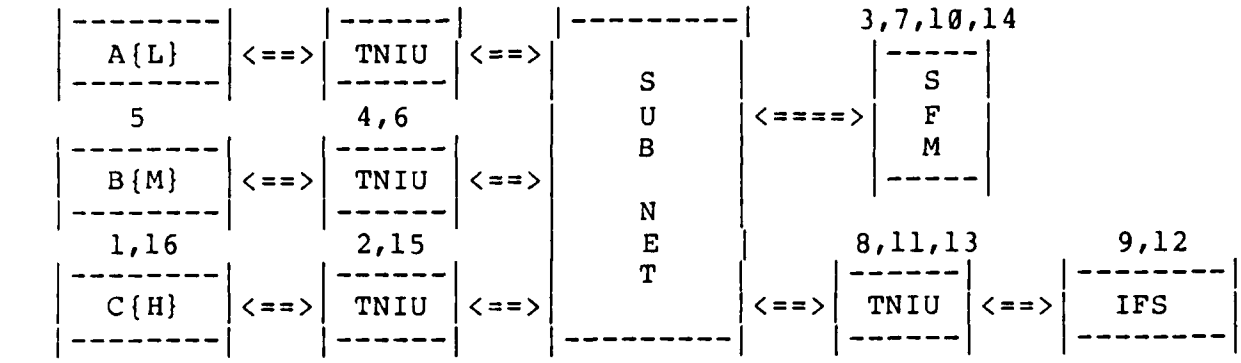


Figure 10. Distributed Secure System Complete

IFS, in order to be truly isolated from the LAN, has a TNIU placed between itself and the subnet. Again, the IFS's TNIU is provided with a special encryption key that is shared only with the SFM's TNIU.

Perhaps the best way to tie all this information together is through a simple illustration (see Figure 11). For example, suppose computer C{H} requires the surface-to-air missile threats for a particular region. Computer C{H} would compose a remote procedure call (RPC) and send it to the SFM. (If the computer maintaining the surface-to-air missile database was in the same security partition as computer C{H}, the two computers would communicate directly, thus avoiding the communication with the SFM altogether. However, for the purposes of this study, computer C{H} is in the level zero partition while computer B{M} is in the level 1 partition.) The SFM examines the RPC sent by computer C{H} to ensure that it is authorized under the existing security policy. Once the identification and authorization functions have been performed by the SFM, the SFM reencrypts the RPC using the key it shares with computer B and then forwards the RPC to computer B{M}. After executing the query on its database, computer B{M} sends the results of the query, after the appropriate processing by both computer B{M} and its associated TNIU, to the SFM. After further processing, the SFM forwards the query response to the IFS. Subsequent to the storage of computer B{M}'s response in the IFS, the SFM reencrypts the file using computer C{H}'s key



1. C composes a RPC requesting SAM data and forwards the RPC to its TNIU.

2. C's TNIU attaches a level Ø label, checksums and encrypts the RPC, and forwards it to the SFM.

3. SFM verifies the message's address, decrypts using C's key, validates the request, attaches a level 1 label, checksums and reencrypts using B's key, and forwards the request to B.

4. B's TNIU verifies the message's address, decrypts and verifies the security label, and forwards it to B.

5. B performs the database operation and forwards the results to its TNIU via a RPC.

6. B's TNIU, after processing, forwards it to the SFM.

7. Similar to (3) except the SFM forwards the response to the IFS, using a key unique to the IFS, under the assumption C cannot receive the response immediately.

8. IFS's TNIU verifies address and forwards response to the IFS.

9. IFS stores response without any processing on the file.

1Ø - 14. When C{H} becomes available, the SFM requests the response file from the IFS. Once it has received it, the SFM validates the file's integrity, attaches a level Ø label, checksums and encrypts the response using C's key, and forwards the file to C.

15. C's TNIU verifies address, decrypts response, verifies the security label, and forwards the response to C.

16. C's application resumes flight path planning.

Figure 11. Database Query Data Flow Diagram

and sends the file to computer C{H}. The question now arises as to why does computer B{M} send its response to the SFM instead of directly to computer A{H}. Besides the obvious reason that computer B{M} does not have the proper encryption key, such actions would violate the logical separation of the communications channels. This separation technique is one of the four upon which this entire design is based. The response to the query must also be completely stored in the IFS prior to the SFM forwarding the response in order not to violate the temporal separation of different encryption keys required for the encryption chip used in the SFM's TNIU. During this entire process, both computers B and C are recording their actions in a local audit trail log.

The final topic considered for the distributed secure system architecture is the internetwork bridge. Connecting the secure LAN to the external communications network is rather straightforward. The bridge consists of a TNIU similar to the one used for computers A{L}, B{M}, and C{H}. All incoming messages and database updates would be sent directly to the SFM once the message has been properly encrypted using a key shared only by the gateway's TNIU and the TNIU associated with the SFM. Once the SFM has properly stored the message in the IFS, the SFM forwards the message to the intended host; again, after the appropriate reencryption had taken place. Messages originating within the LAN would be handled in a similar manner, simply

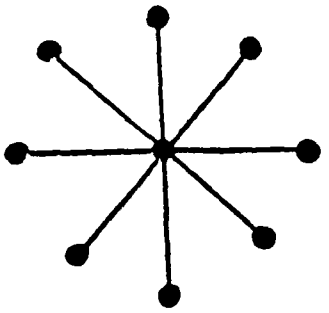
reversing the process.

In conclusion, this section has introduced four secure LAN architectures. Each one was presented in a manner which made no presumptions about the particular local area network topology being used. The next section of this chapter presents several topologies and examines the tradeoffs involved when each of these topologies is implemented in a secure environment. Subsequent to this section, a complete summary of this chapter is accomplished resulting in the recommendation of a LAN architecture and topology which provides the required operationally secure environment as specified by this thesis' sponsor.

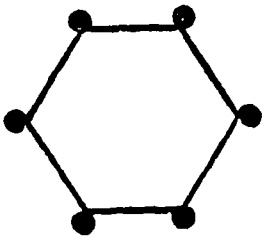
LAN Topologies

As the development of local area networks has become more commonplace, the number of different LAN topologies has grown. In order to limit this discussion, the following four common topologies are analyzed: linear bus, star, ring, and radio broadcast (see Figure 12).

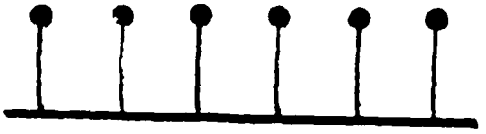
Linear Bus. The linear bus topology is the most widely employed in LANs. Typically, either a baseband or a broadband bus is utilized in this topology. In either case, the configuration uses a single coaxial cable or dual cables which serve as the main trunk of the network. Network subscribers are attached to the main trunk either directly via a smaller cable which extends to the user location or indirectly via an alternate trunk branching out from the main trunk. Bus interface units (BIUs) provide access to



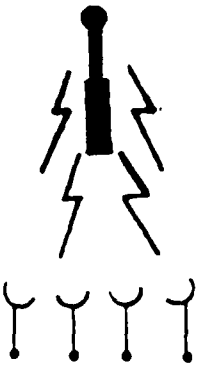
STAR



RING



LINEAR BUS



RADIO BROADCAST

Figure 12. Topologies

the network and all the BIUs receive the network traffic simultaneously but ignore those not specifically addressed to them.

The baseband variation of the linear bus topology is based on a coaxial cable which has high propagation velocity and low transmission losses at small distances. The network size is limited to about 3 kilometers due to insertion loss where spurs are attached to the main bus. This problem can be overcome through the use of bridges which link together multiple baseband networks over a large geographical area. On the other hand, the broadband bus is based on community antenna television technology using radio frequency transmission. The use of frequency division multiplexing allows the simultaneous use of multiple carrier frequencies. A broadband system also provides the ability to connect nodes to the bus and allows splitting of signals along multiple lines rather than configuring them to a single cable backbone as in the baseband system. Both the baseband and the broadband systems usually control access to the cable by a contention algorithm incorporating carrier sense multiple access with collision detection. Although the broadband bus provides greater throughput and larger geographic distribution, the baseband is much less expensive for networks with only a few subscribers and requires a much smaller initial investment. However, the advantages of the broadband system include both greater flexibility and expandability. But most importantly from a security point

of view, the broadband system allows for multiple channels, with each channel being used for a different security partition.

Star. The star topology consists of a central switching processor which provides interconnection for all the LAN components. Each node is connected to a central point where tandem switching is carried out to interconnect outlying nodes by trunk groups. Perhaps the greatest disadvantage of this topology is its vulnerability should the central node become disabled. To a lesser extent, the central node provides a potential bottleneck should LAN traffic become sufficiently active. However, this topology does provide security advantages not offered by the others discussed here. For instance, each node within the LAN is connected to the central node by a physically different cable which provides some measure of physical separation of the communications over the LAN. Additionally, the central node would be an ideal location for the reference monitor function required by most of the architectures described in the previous section. Finally, the star topology's central node would be a suitable location from which to establish an internetwork bridge. Obviously, the above mentioned advantages would only be realized if the LANs workload was such that it would not overload the central node.

Ring. The ring topology is one in which all LAN resources share a common communication bus which is closed upon itself. Because of its structure, each transmission

returns to the originator, delayed by some amount dependent upon the propagation delays of the network components. Messages which are not addressed to a the interface unit currently receiving it are then retransmitted to the next interface unit in the same sequence. Moreover, a vulnerability common to not only the ring but also both the linear bus and the star is that these topologies are highly susceptible to unauthorized traffic analysis; however, as has been mentioned before, this study assumes a nonmalicious environment. Physical line loss vulnerabilities can be overcome by providing for backup paths throughout the ring. The advantages of the ring topology include the relatively low implementation cost of the interface between the hosts and the subnet and the simple communications technology which utilizes only digital devices. Whether the low cost is actually realized, particularly after the security enhancements some of the aforementioned architectures require, is still an unanswered question. Similar to the star topology, one node in the ring could act as a central reference monitor since LAN traffic passes by all the network nodes.

Technically, the ring topology has two variations. The first, referred to as a loop, passes messages between the nodes in their entirety. Each node in the loop stores the message until the required output line is free to transmit the message to the next node in the scheme. The second variation, referred to specifically as a ring, transmits

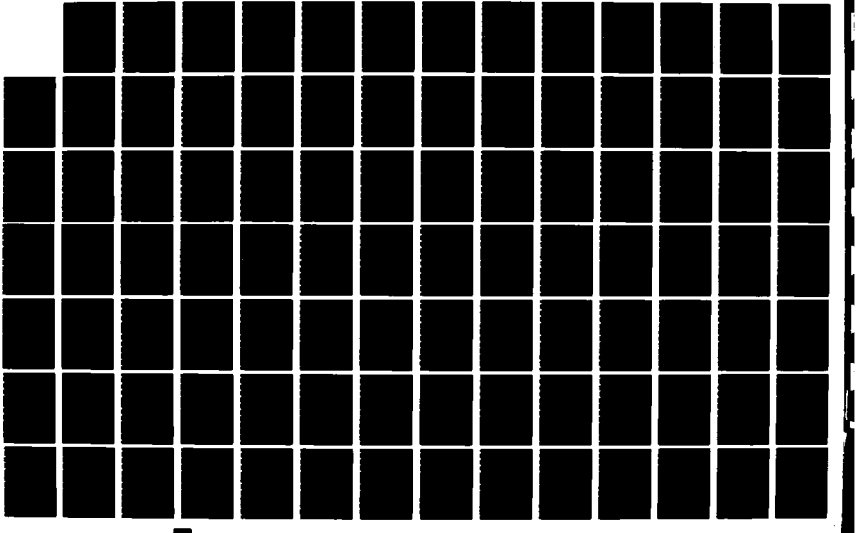
each bit of the message propagating around the ring on its own, not waiting for the rest of the message to which it belongs. The advantages and disadvantages of this topology apply to either of these transmission schemes.

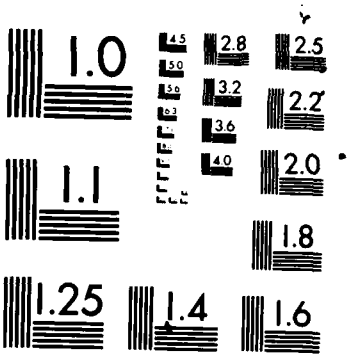
Radio Broadcast. Similar to the ring topology, radio broadcast networks allow each node to view every message and to contend for a common communication resource. Differences from the ring network include the ability to have separate, asymmetric data rates for the two sides of a dialog and the way in which requestors attempt to gain control of the subnet. Rather than using a contention based algorithm, the radio broadcast topology uses a "transmit and see if it gets through scheme." The lack of available frequency spectrum and geographical coverage problems restrain the usage of this particular topology. Although the geographical coverage problem is not relevant to the problem this study is trying to resolve, the frequency problem narrows the bandwidth which potential "tappers" of the LAN would have to search. In other words, this topology is vulnerable to unauthorized use of the LAN, thereby introducing severe denial of service threats along with the potential for jamming.

Each of the local area network topologies just described provides the secure LAN designer several advantages and disadvantages to consider. Of the four topologies discussed, perhaps radio broadcasting offers the greatest disadvantages when considered in the environment described

AD-A172 787 ANALYSIS AND DESIGN OF A MULTI-SECURITY LEVEL DISTRIBUTED DATABASE SYSTEM(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING 2/3

UNCLASSIFIED R A MOELLER DEC 85 AFIT/GCS/ENG/85D-10 F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

for the purposes of this study. Each of the remaining three provide relatively similar security advantages, although the star topology provides a greater degree of physical communication security because each message is not examined by every other node in the network. However, the vulnerabilities of the central node in the star requires a more complex redundancy scheme should the central node fail. When the advantages and disadvantages are considered strictly in relation to the thesis' operation environment, this author feels that either the ring or star topological configurations are appropriate.

Conclusions and Summary

The primary emphasis of this chapter has been to describe alternative secure local area network architectures. The architectures presented included a multiple channel network, trusted interface units, a security controller, and a distributed secure system. Before identifying the specific architecture considered the most appropriate, a discussion of the reasons behind the decision was presented.

The computer network security problem is not merely a communications problem, but rather a complex set of problems that are due to the multi-system structure of LANs. In effect, the network environment adds a new dimension to the multi-resource problem of a single system; thus adding an additional level of complexity to a problem which was already quite complex. The solutions to the security

aspects of this thesis' problem could be applied at any of the several layers at which the problem exists. For instance, solutions could have been proposed at either the database management system, operating system, or network levels. In the first three architectures examined in this chapter, the designs used a multiple layer approach. These architectures provided some security mechanisms at the network interface level. But most importantly, each of the first three architectures made the assumption that each host's operating system was trustworthy. Perhaps, some time in the future, this assumption will be attainable but for a design which is going to be implemented in the near future, such an assumption is unrealistic. The distributed secure system, when contrasted with the previous architectures, provides a much more realistic and attainable approach. The only assumption the DSS made in regards to each host's operating system was that it was untrustworthy, which is probably closer to reality, given the current state of software verification techniques.

Although several other issues, such as hardware and software costs, LAN complexity, and system performance, could be addressed in detail, this discussion stresses what this author feels are the two most important issues. The first of these issues is feasibility. Each of the architectures present the LAN designer with sound approaches to the security problem. If the decision were to be based upon feasibility alone, the designer would have a tough time

deciding. However, as was briefly discussed above, the second issue is the timeliness of the proposed architecture's implementation. The DSS is proposed with all the necessary tools for its implementation available today. The multiple channel network, trusted interface units, and security controller architectures depend on advancements to be made in the art of software verification techniques. As was discussed in Chapter III, current software verification tools are not capable of providing the necessary assurances for large, complex computer programs such as computer operating systems. Therefore, as the goal of this chapter was to propose a secure local area network architecture which could be implemented in the near future, the obvious choice is the distributed secure system.

Naturally, the choice of the DSS involves many tradeoffs particularly with respect to total LAN cost and complexity. The truly incremental costs of the DSS relative to the other architectures presented is the cost of the host-level isolated file store. Yet the IFS's cost would be much lower than the cost of the additional host-level machine required in the security controller architecture. The issues of system complexity and performance degradation remain unanswered. This author believes the DSS performance factor would be no worse than the degradation currently being experienced on systems which are being implemented with the addition of a security kernel.

Finally, the last topic considered in this chapter

involved the different topological configurations being implemented in local area networks. Again, either the ring or star topologies would be appropriate for the needs of this study.

V. Database Design

Introduction

The database design process involves two separate, yet interrelated, stages. The first stage begins with an examination of the parameters described in Chapter II. In order to provide a better understanding of the kinds of information the database is supposed to store, these parameters are presented using the informal entity-relationship model. During the second and final stage of the design process, the database's conceptual scheme, as depicted through an entity-relationship diagram, is translated into the relational data model.

Entity-Relationship Model

The entity-relationship model (ERM) provides an easy-to-understand means to explore not only the various individually distinguishable things within the database, referred to as entities, but also the relationships which exist between these entities. However, before presenting the ERM of this thesis' database, a few terms used in the model must be thoroughly defined.

The notion of an entity is all encompassing. Simply put, an entity is something which exists and can be distinguished from other, possibly nondistinctive objects. For example, an aircraft is an entity. So is a missile site. The grouping of "similar" entities into entity sets is one of the primary steps in the modeling of the database.

All entities have properties which describe the object in detail. These properties, or attributes, associate a value from a domain of values for that attribute with each entity in an entity set (5:13). A key is the attribute or set of attributes which uniquely identifies each entity within a given entity set. The term relationship, as formally defined by Ullman, is simply an ordered list of entity sets. For instance, suppose SITES and MISSILES are two entity sets. Also assume that the relationship set LOCATED_AT describes the relationship between SITES and MISSILES. Each tuple of the form (e_1, e_2) in the set LOCATED_AT implies that e_1 is in SITES and that e_2 is in MISSILES. The last term to be addressed, and perhaps the most important, is functionality.

Functionality concerns the classification of relationships between entities according to how many entities from one set can be associated with the entities of a second set. Typically, the relationships between entity sets are classified as either one-to-one, one-to-many, or many-to-many. Continuing with the SITES and MISSILES example, suppose each missile type could only be located at a specific site and that this site could only have the aforementioned missiles located there. Thus, the relationship set LOCATED_AT would describe a one-to-one relationship. However, if more than one missile type could be located at a specific site, than the LOCATED_AT relationship set would depict a one-to-many relationship. Finally,

assume not only that a site may contain more than one missile but also that a missile may be deployed to more than one location. In this last instance, the LOCATED_AT relationship set now describes the more complex case of a many-to-many relationship.

In modeling the database, a diagrammatical tool, which greatly simplifies both the author's manner of presentation and the reader's ability to more fully understand what it being presented, is used. This tool, known as the entity-relationship diagram, uses a variety of symbols to graphically describe the database being designed. One of the most important symbols used is the rectangular box, which depicts entity sets. Equally important are the diamond and elliptical shapes. The diamonds represent relationship sets while the elliptical symbols depict both entity set and relationship set attributes. The entity-relationship diagram also depicts the functionality between entities through the use of numbers or variables, as annotated along the vectors leading into or out of the relationship sets.

The complete conceptual design of the database is presented in Figures 13-16. Basically, the entity sets were those presented in Chapter II. The two principal entity sets, not previously described, include MISSION and AIRCRAFT.

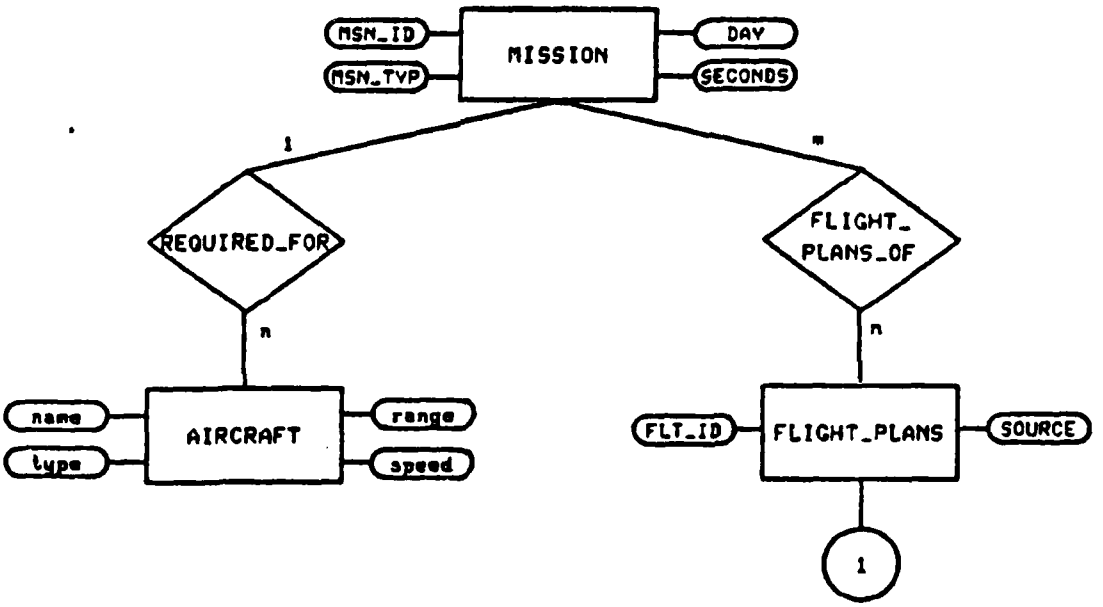


Figure 13. Conceptual Database Design - Mission

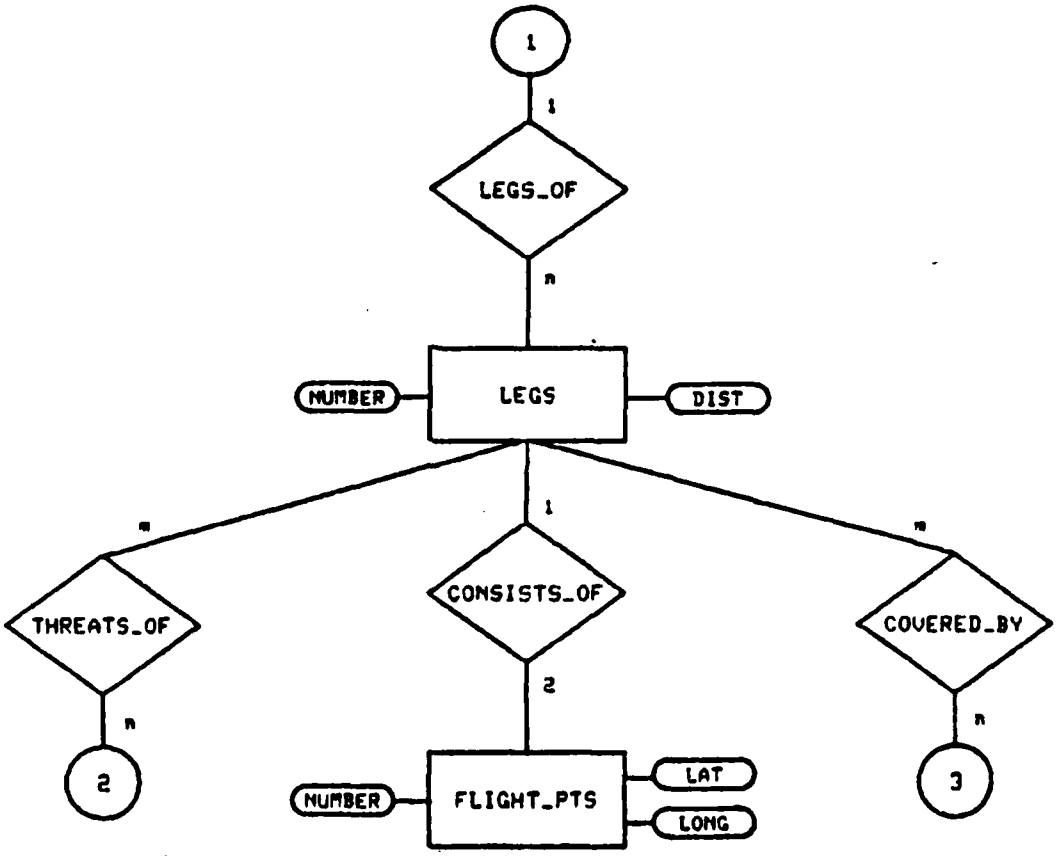


Figure 14. Conceptual Database Design - Legs

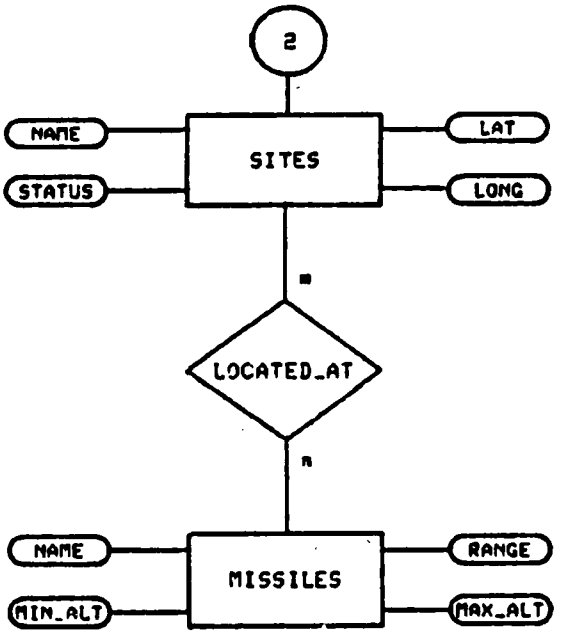


Figure 15. Conceptual Database Design - Sites

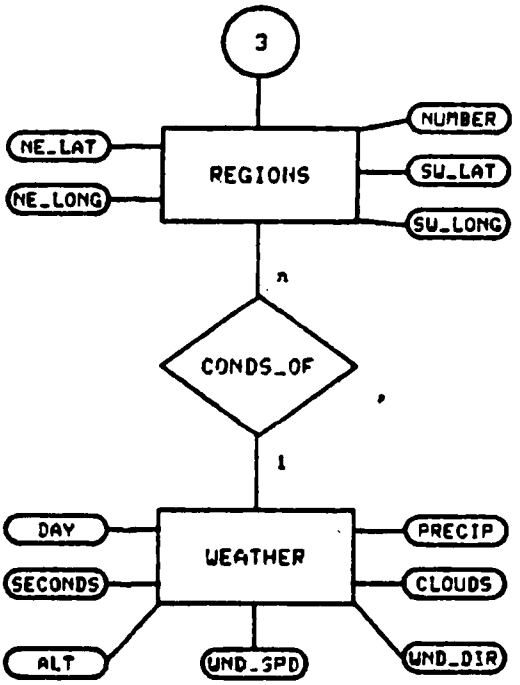


Figure 16. Conceptual Database Design - Regions

Relational Data Model

Of the three predominant data models in use today, the relational data model was preferred for several reasons over both the hierarchy and network models. The most important reason for selecting the relational model was the high degree of data independence this model provides. Regardless of the application being developed now or being considered for the future, the relational model provides a sound basis which should not require the rewriting of existing programs just because the physical implementation of the conceptual scheme by the physical scheme has changed. Another important reason for selecting the relational model was the relative ease of understanding relations. The number of basic constructs within this model is just the relation itself, a construct which is both simple and familiar. Furthermore, the number of distinct operations is small because of having only one data construct to deal with. Finally, the data manipulation language used within the relational model provides for "symmetric exploitation", which is defined by Codd to be the ability to access a relation by specifying known values for any combination of its attributes, seeking the unknown values for its other attributes (17:478).

The transformation of the entity-relationship diagram, as presented in the previous section, into the relational model is rather straightforward. Each entity set is mapped directly into a relation, with each of the entity set's

attributes becoming attributes of the relation. The relationship sets are also mapped into relations, taking as attributes the key attributes of each of the associated entity sets. Figures 17-20 depict the results of the entity-relationship diagram to relational data model transformation. The relations are presented in these figures according to the particular computer system each relation is maintained on, either A{L}, B{M}, or C{H}.

Conclusion

The design of the thesis' database, both conceptually and physically, was presented in this chapter. The use of both the entity-relationship model and the relational model greatly simplified this task. One topic not specifically addressed before involves the details of the algorithm used in partitioning the database amongst the various nodes of the local area network. This was not an intentional oversight but rather an omission of discussing a problem made trivial due to nature of the network designed in Chapter IV. As a final note, each relation is classified according to the level of sensitivity of the computer on which it is maintained. For example, all relations stored on computer B{M} are classified as level one relations, likewise for computer A{L} and computer C{H} relations.

REGIONS

NAME	NE_LAT	NE_LONG	SW_LAT	SW_LONG
LONDON	55.0	5.0	50.0	0.0
BERLIN	60.0	15.0	55.0	10.0
LENINGRAD	60.0	28.0	56.0	20.0

CONDS_OF

REGION NAME	DAY	SECONDS	ALT
LONDON	91055	00001	10000
BERLIN	91055	5000	50000
LENINGRAD	91055	18000	500

WEATHER

DAY	SECONDS	ALT	WND_SPD	WND_DIR	CLOUDS	PRECIP
31055	00001	10000	5	30.0	C	N
91055	5000	50000	25	30.0	B	R
91055	18000	500	35	210.0	X	S

Figure 17. Computer A(L) Relations

SITES

NAME	LAT	LONG	STATUS
MOSCOU	55.8	37.5	0
LENINGRAD	60.0	30.5	0

LOCATED_AT

SITE_NAME	MISSILE NAME
MOSCOU	SA-3
LENINGRAD	SA-5

MISSILES

NAME	MIN_ALT	MAX_ALT	RANGE
SA-3	1000	43000	18500
SA-5	5000	95000	185000

Figure 18. Computer B(M) Relations

MISSION

MSN_ID	MSN_TYP	DAY	SECONDS
ROLLING THUNDER	BOMBING	91055	0001
LINEBACKER	BOMBING	91055	3600

REQUIRED_FOR

MSN_ID	AIRCRAFT NAME
ROLLING THUNDER	B-1B
LINEBACKER	FB-111A

FLIGHT_PLANS_OF

MSN_ID	FLT_ID
ROLLING THUNDER	1A
LINEBACKER	1B

LEGS_OF

FLT_ID	LEG_ NUMBER
1A	1
1A	5
1B	2

AIRCRAFT

NAME	TYPE	SPEED	RANGE
B-1B	SB	650	7500.
FB-111A	TB	900	4100
F-15	F	1100	2800

FLIGHT_PLANS

FLT_ID	SOURCE
1A	UPPER HEYFORD
1B	LAKENHEATH

Figure 19. Computer C(H) Relations

LEGS

NUMBER	DISTANCE
1	1500
2	1750

THREATS_OF

LEG_ NUMBER	SITE_ NAME
1	LENINGRAD
2	MOSCOW

COVERED_BY

LEG_ NUMBER	REGION_ NUMBER
1	3
2	5

CONSISTS_OF

LEG_ NUMBER	FLIGHT_ PTS_#
1	1
1	2
2	3
2	4

FLIGHT_PTS

NUMBER	LAT	LONG
1	59.0	28.0
2	60.0	30.5
3	52.0	24.0
4	55.8	37.5

Figure 20. Computer C(N) Relations (continued)

VI. Implementation

Introduction

A partial implementation of the database designed in the previous chapter was successfully accomplished. Although the implementation was simplified for this thesis, it provides a strong argument for the validity of the local area network proposed to support the multi-security level database which was provided by Headquarters, Space Division. The specific objectives attained in this phase of the thesis effort were threefold. First, the effects of data encryption on overall database performance were found to be minimal. Secondly, the additional network overhead necessitated by the central position of the secure file manager was found to be substantial but its effects were not so burdensome as to render this design unmanageable. Finally, the total lines of code required for the construction of the secure file manager was found to be within the realm of currently available software verification techniques. However, no attempt was made to prove the correctness of any of the programs written for this project. Each of these objectives are discussed more fully in the remainder of this chapter, but before proceeding with this discussion, the local area network architecture and the computer software's configuration are presented.

Local Area Network Architecture

Figure 21 shows a pictorial representation of the architecture used to implement the network designed in Chapter 4. The computer architecture chosen for this partial implementation consists of two S-100 microcomputers running dBASE II; two LSI-11 microcomputers connected into the local area network via the Network Operating System (NETOS) (2:1), where System L is a front end to computer A{L} and System S is a front end to computer B{L}; three LSI-11 microcomputers, Systems C, D, and K, which act as computer C, Secure File Manager, and Isolated File Store; and a final LSI-11, System B, acting as the central system in NETOS, through which all encrypted network messages are passed.

The implementation described here deviates from the original design in two ways. First, due to the limited resources of the LSINET, none of the systems are connected to network through the use of "trusted" network interface units. (The software implications of this are discussed in the next section.) The final deviation from the original design concerns the interface to the external communications network. Although the facilities of the LSINET could simulate such an interface, it was not attempted in this implementation.

Software Configuration

The computer software used for this study can be divided into four major categories: operating system software, network software, application software, and database manage-

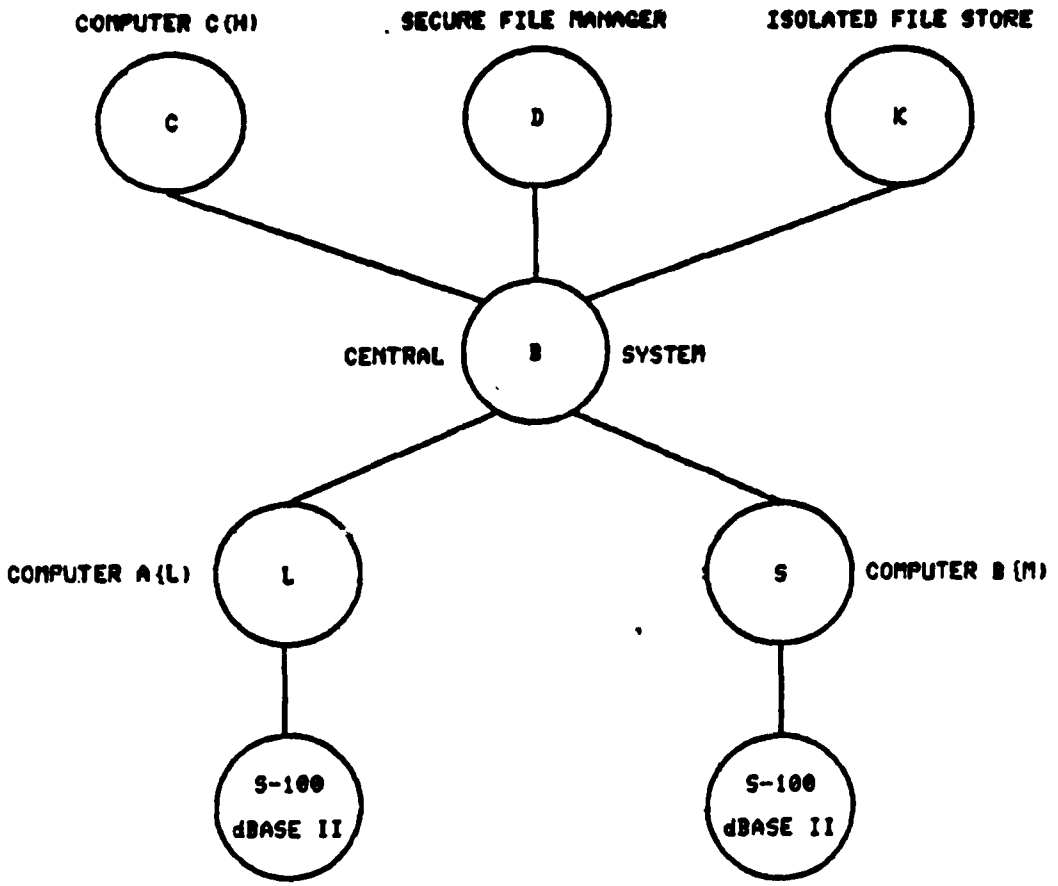


Figure 21. Implementation Architecture.

ment system software. However, as was mentioned before, no attempt was made at proving program correctness for any of these categories.

The operating system software used on the LSI-11s was Digital Equipment Corporation's (DEC's) RT-11 Version 5.01 C whereas the S-100s used CP/M Version 2.2. As was discussed in Chapter 4, the trustworthiness of each host's operating system was insignificant. However, the Secure File Manager, which is an enhanced "trusted" network interface unit in the design, was implemented on an LSI-11 and uses the facilities provided by its operating system. Although this is a major deviation from the original design, this implementation was not intended to provide an operational model. Furthermore, this deviation does not substantially detract from the original objectives of the implementation.

The network software (NETOS) used by this study was developed through course work and special student projects at the Air Force Institute of Technology. The NETOS incorporates the International Standards Organization's (ISO's) Reference Model of Open Systems Interconnection (2:6). This software was used as developed except for the modifications made to layer 5 of the ISO model to allow for encryption. The encryption algorithm chosen was simply one of exclusive or'ing each byte with the system's key as it was either received from or transmitted to the network. All of the systems which interfaced with the network, with the exception of the central system, had its own key with which

it communicated with the secure file manager (SFM). The session layer of the SFM, layer 5, was modified so that it could both encrypt and decrypt the information it received, regardless of the information's source.

The application software which was used in this project was developed by a number of sources, although this author tailored it to the specific needs of the study, given the limited time which remained to provide even a simplified implementation. Perhaps the most significant deviation from the design was the manner in which the database queries were made. As implemented, only computer C{H} queries the other two computers. Computer C{H}'s application was written so that the user interactively requests that a specific dBASE II command file is sent to either computer A{L} or computer B{M}. While the request is being serviced, the user is prohibited from performing any further actions. Once the database request has been satisfied, the user is prompted to determine if he would like to view the results. There is not a flight path application, as described in Chapter 2, which would use the response to the query once it had been received by computer C{H}. Both computer A{L}'s and B{M}'s application were also greatly simplified. These two computers simply wait to receive database queries from computer C{H} via the SFM. Once the query file has been received and decrypted, the file is downloaded to the S-100s for dBASE II execution. While the file is being executed on by the S-100, the user is prohibited from further actions.

Once the query has been serviced, the response file is uploaded to its respective host, encrypted, and forwarded to computer C{H} via the SFM.

The most important software written for this thesis was the Secure File Manager. Perhaps the greatest task of this program was to ensure that only the proper query file was received and then forwarded to its true destination. Although all of the program listings are available in Volume II, a brief description of one of the Secure File Manager's most important modules, Val_Transfer, highlights some of the tasks this software must be proven to perform correctly.

Before computer C{H} could send a query to either of the other two host computers, it first had to request permission from the SFM. The contents of this request included the identifier of the sending computer, the name of the command file to be sent, and the size of the command file. Once the SFM had received these three parameters, the module Val_Transfer would ensure that the sending computer had the authority to be making such a request. Once the request had been approved, the SFM would notify computer C{H} that it could forward the query. Again, after the SFM had received the actual command file, Val_Transfer would check the command file's size and a label, internal to the command file, to ensure that it had received the proper query. If either the file's size or label did not match with what the request had provided, the SFM notifies computer C{H} and the

transaction is terminated. Additionally, Val_Transfer checks a second label internal to the command file to ensure that the query is being made on the same computer as indicated with the network control information received with the file. Again, if any discrepancies exist, computer C{H} is notified and the transaction is terminated. These tasks are but a few of the checks which must be built into the code in order for the secure file manager to properly perform its role in multi-security level distributed database problem.

The final software category concerns the database management system (DBMS); which, for the purposes of this study, is the relational DBMS dBASE II. Both computer A{L} and computer B{M} in the implementation act as front ends to the actual database machines which, as was depicted in Figure 21, are S-100s. Once either computer A{L} or computer B{M} has received a query (command file) from the SFM, it automatically downloads the query to the S-100 for execution. The relations upon which dBASE II is executing are presented in Figure 22. These relations were constructed in accordance with the design presented in Chapter 5. Obviously, only the relations which reside on either computer A{L} or computer B{M} were included for data manipulation by the command files used in this implementation. After the query had been downloaded and executed by dBASE II, the S-100 notifies the front end, either computer A{L} or B{M}, which in turn uploads the results and forwards

```
STRUCTURE FOR FILE: SITES.DBF
NUMBER OF RECORDS:  00002
DATE OF LAST UPDATE:  09/19/85
PRIMARY USE DATABASE
FLD  NAME    TYPE WIDTH  DEC
001  NAME    C    010
002  LAT     N    006    003
003  LONG    N    007    003
004  STATUS  C    001
** TOTAL **       00025
```

```
STRUCTURE FOR FILE: MISSILES.DBF
NUMBER OF RECORDS:  00002
DATE OF LAST UPDATE:  09/19/85
PRIMARY USE DATABASE
FLD  NAME    TYPE WIDTH  DEC
001  NAME    C    005
002  MINALT  N    006
003  MAXALT  N    006
004  RANGE   N    006
** TOTAL **       00024
```

```
STRUCTURE FOR FILE: LOCATED.DBF
NUMBER OF RECORDS:  00002
DATE OF LAST UPDATE:  09/19/85
PRIMARY USE DATABASE
FLD  NAME     TYPE WIDTH  DEC
001  NAME     C    010
002  MISNAME  C    005
** TOTAL **       00016
```

```
STRUCTURE FOR FILE: REGIONS.DBF
NUMBER OF RECORDS:  00003
DATE OF LAST UPDATE:  09/19/85
PRIMARY USE DATABASE
FLD  NAME    TYPE WIDTH  DEC
001  NAME    C    010
002  NELAT   N    006    003
003  NELONG  N    007    003
004  SWLAT   N    006    003
005  SWLONG  N    007    003
** TOTAL **       00037
```

```
STRUCT FOR FILE: CONDS_OF.DBF
NUMBER OF RECORDS:  00003
DATE OF LAST UPDATE:  09/19/85
PRIMARY USE DATABASE
FLD  NAME     TYPE WIDTH  DEC
001  RGNNAME  C    010
002  DAY      N    005
003  SECONDS  N    005
004  ALT      N    006
** TOTAL **       00027
```

```
STRUCTURE FOR FILE: WEATHER.DBF
NUMBER OF RECORDS:  00003
DATE OF LAST UPDATE:  09/19/85
PRIMARY USE DATABASE
FLD  NAME    TYPE WIDTH  DEC
001  DAY     N    005
002  SECONDS N    005
003  ALT     N    006
004  WNDSPD  N    003
005  WNDDIR  N    007    003
006  CLOUDS  C    001
007  PRECIP  C    001
** TOTAL **       00029
```

Figure 22. Implementation dBASE II Relations Structure.

them to the SFM. As was the case for all the previous software categories, no attempts were made at proving program correctness for the DBMS. However, in both the proposed design and its implementation, the trustworthiness of the DBMS or lack there of, has no bearing on the security aspects of the problem.

Testing and Results

This section describes the tests which were performed on the network (hardware and software) just described. Although these tests are not rigorously defined, their results provide this author's optimism on the verifiability of the solution provided by this thesis to the security problem as described in Chapter I. For all phases of testing, the dBASE II command files presented in Figures 23 and 24 were used and included two join commands, typically time consuming operations.

The first objective of determining the impact of the encryption algorithm on system performance was tested by querying the database on computer A{L} with the encryption flag in layer 5 of all programs turned off (see Volume II for coding details.) In other words, the network was performing normally except that all files were being transmitted in clear text. After executing the command file three times, the encryption flag in layer five was turned on and computer A{L} was queried three more times.

```
*CA
*****************************************************************
*                                                               *
*     DATE:  09/16/85                                           *
*     VERSION: 1.0                                              *
*                                                               *
*     TITLE:  Computer A's dBASE II Query File                  *
*     FILENAME:  TESTA.CMD                                      *
*     COORDINATOR:  Ron Moeller                                 *
*     PROJECT:  Thesis                                          *
*     OPERATING SYSTEM:  CP/M 2.2                               *
*     LANGUAGE:  dBASE II Application Language                  *
*     USE:  This command file is sent from Computer C           *
*           to Computer A via the SFM to test the               *
*           application programs written for the project        *
*           developed by the above mentioned coordinator.       *
*                                                               *
*     NOTE: The first line in this file indicates the           *
*           source computer and the destination computer.       *
*           This line must be included in all command           *
*           files because this information is validated         *
*           once it is received by the SFM.                     *
*                                                               *
*****************************************************************
set talk off
set default to b:
set alternate to dbresult
set alternate on
use regions
select secondary
use conds_of
join to templ for p.name = s.rgnname .and. p.name = "London" ;
     field name,day,seconds,alt
use templ
select secondary
use weather
join to temp2 for p.day = s.day .and. p.seconds = s.seconds ;
     .and. p.alt = s.alt field name,day,seconds,alt,wndspd, ;
     wnddir,clouds,precip
use temp2
display all off
set alternate off
clear
delete file templ
delete file temp2
quit
```

Figure 23. TestA dBASE II Command File.

```
*CB
*****************************************************************
*                                                               *
*     DATE:  09/16/85                                           *
*     VERSION: 1.0                                              *
*                                                               *
*     TITLE:  Computer B's dBASE II Query File                  *
*     FILENAME:  TESTB.CMD                                      *
*     COORDINATOR:  Ron Moeller                                 *
*     PROJECT:  Thesis                                          *
*     OPERATING SYSTEM:  CP/M 2.2                               *
*     LANGUAGE:  dBASE II Application Language                  *
*     USE:  This command file is sent from Computer C           *
*           to Computer B via the SFM to test the               *
*           application programs written for the project        *
*           developed by the above mentioned coordinator.       *
*                                                               *
*     NOTE: The first line in this file indicates the           *
*           source computer and the destination computer.       *
*           This line must be included in all command           *
*           files because this information is validated         *
*           once it is received by the SFM.                     *
*                                                               *
*****************************************************************
set talk off
set default to b:
set alternate to dbresult
set alternate on
use sites
select secondary
use located at
join to temp1 for p.name = s.name .and. p.name = "Moscow" ;
     field nam,misname
use temp1
select secondary
use missiles
join to temp2 for p.misname = s.name field name,minalt, ;
     maxalt,range
use temp2
display all off
set alternate off
clear
delete file temp1
delete file temp2
quit
```

Figure 24. TestB dBASE II Command File.

Table I. Testing Results

	Clear Text			Encrypted		
Start:	21:02:00	21:06:52	21:17:53	21:24:09	21:28:11	21:33:10
Stop :	21:05:08	21:10:03	21:21:02	21:27:20	21:31:20	21:36:23
Time :	3:08	3:11	3:09	3:11	3:09	3:13

The average times for the clear text and encrypted transmissions were 3 minutes, 9.33 seconds and 3 minutes, 11 seconds respectively. With an average time difference of only 1.67 seconds, it is obvious that encrypting the data had little impact on system performance. Moreover, an operational implementation of this design would use an encryption algorithm encoded on an integrated circuit; thus, further improvements in performance would be expected.

The second objective of determining the impact of the additional network message traffic necessitated by the fact that all inter-host communications must be monitored by the secure file manager was tested. Again, using the results of the previous tests of the computer A{L} database, it was observed that it took, on the average, 1 minute, 48 seconds from the time computer A{L} had received notification of a pending query to the time computer A{L} had completely forwarded the results of the query back to the SFM. As was mentioned before, total execution time from the moment computer C{H} had notified the SFM to the time computer C{H} had completely received the results file was 3 minutes, 11 seconds. Thus, approximately 43% of the total execution time was used for the additional message traffic required by

this network design. Although this may seem substantial, the dBASE II command files used for testing, as well as the database itself, were rather simple in nature. Therefore, as the complexities of the queries increase and the database becomes larger, the time for query execution will increase, which will, in turn, reduce the percentage of time spent on network traffic. However, one fact, not previously mentioned, is that all testing was performed on a dedicated network. In other words, there was no other traffic flowing on the LSINET during the testing periods. In a busy network environment, total query execution times would surely be greater than the times recorded during the limited testing performed at this time.

The final objective analyzed during the testing phase of this thesis project involved the size of the secure file manager program which was developed. The total number of lines of source code for this program was approximately 1250 lines. Of these 1250 lines, approximately 500 lines were comments. Given the nature of the development cycle for this project, another 250 lines of code could be removed by improving the SFM's design. Therefore, even after enhancing the security checking performed within this program, the Secure File Manager would be well under 1000 lines of code. Thus, this program would be of a small enough size to be formally proven using the automated software verification techniques discussed in Chapter 3 (13:95).

Conclusion

The operational implementation of a local area network which supports a multi-security level distributed database is attainable today as evidenced by this thesis' effort at a simplified version of the LAN designed in Chapter 4. Although a number of simplifications were included in this first implementation attempt, the attainment of the above mentioned objectives makes this solution worthy of further investigation, particularly in the area of proving software correctness. Hopefully, as the LSINET matures and more computing resources become available, a complete implementation of the LAN will support this author's optimism on the validity of his solution.

VII. Conclusion

Summary

The analysis and design of a multi-security level distributed database was a complex task. During the course of the development of the solution presented in this thesis, several areas within the field of computer science were discussed. During the analysis of the problem, much research into the present state of the art of software verification techniques was accomplished. This research directly affected the nature of the solution being sought in that if a solution other than the typical "system high" approach was to be achieved, the problem would have to divided into small enough parts so that each of the part's function would be within the realm of contemporary software verification techniques.

Once this fact had been realized, the thesis effort was directed toward possible solutions within the total local area network environment, rather than a single system implementing all the required security mechanisms. Again, researching available LAN security mechanisms resulted in the discovery of the distributed secure system being developed by two professors at the University of Newcastle upon Tyne in Great Britain. Applying the specific requirements of this project to the general LAN environment described by Professors' Rushby and Randell, provided an excellent solution to the problem being investigated. Having decided on a specific design, the next step

undertaken was a partial implementation. The specific objectives which were successfully achieved by this work included the fact that data encryption has little or no impact on system performance, that the impact of the additional overhead necessitated by the secure file manager on system performance was substantial yet not of a sufficient enough nature to warrant the disregarding of this approach, and that the SFM's size, as written in the "C" programming language, was within the realm of contemporary software verification techniques. Thus, it is this author's opinion that the distributed secure system, as applied in this particular environment, is a feasible solution to the problem described in this thesis.

Recommendations

The initial scope of this thesis was to analyze and design a multi-security level distributed database. However, as the research progressed, it became clearer as to the exact nature of the problem, particularly, if the solution found was to be fully implemented within the near future. The scope of the redefined problem and the time limitations with which all thesis projects are placed under necessitated cursory examination of certain aspects of this problem.

With the above statements in mind and from the experiences of this author gained throughout the project, the following recommendations, both general and specific,

are made:

(1) Examine thoroughly the intricate details of formal software verification techniques and, if the tools are available, apply these techniques to a specific program.

(2) Investigate contemporary secure operating systems and provide an analysis of the various systems operational capabilities, particularly for those systems used in a military environment.

(3) Implement a complete secure file manager, which has undergone a rigorous proof of correctness.

(4) Perform a formal analysis of a full implementation of the system presented in this thesis, to include a complete application package and the integration of "trusted" network interface units.

(5) Design and implement a secure file manager which would run on a "bare bones" machine. In other words, the program would have to be written so that any operating system facilities, which were used in the first implementation, are not available and would have to be encoded in the new version. Once this has been accomplished, encode the software on a programmable, read-only memory and test its performance against previous versions.

(6) Integrate the security mechanisms attributed to the distributed secure system into a database system which entails distributed processing as well as distributed data. Again, subject the redefined secure file manager to formal software verification techniques.

Final Comment

One of the central themes of this entire effort has been being able to decide whether or not a certain piece of software can be "trusted." Verifying program correctness, or having the ability to prove that a computer program will do exactly what it is designed to and nothing else, is a complex undertaking for any program not of a trivial nature. The importance of this task is being realized in all aspects of computer science and rightfully so. As more complex tasks are being resolved through the use of computers, the science of proving program correctness as well as the tools which facilitate the accomplishment of this task, will become increasingly important. In the military environment, where human life is at stake or the security of our nation is involved, nothing less than absolute, verifiably correct software should be trusted.

AFIT/GCS/ENG/85D-7

STRUCTURE CHARTS FOR THE DISTRIBUTED SECURE SYSTEM

APPENDIX A

TO

THE ANALYSIS AND DESIGN

OF A

MULTI-SECURITY LEVEL DISTRIBUTED

DATABASE SYSTEM

RONALD A. MOELLER
CAPT USAF

Graduate Computer Systems
13 December 1985

AFIT/GCS/ENG/85D-7

USER'S GUIDE FOR THE DISTRIBUTED SECURE SYSTEM

APPENDIX B

TO

THE ANALYSIS AND DESIGN

OF A

MULTI-SECURITY LEVEL DISTRIBUTED

DATABASE SYSTEM

RONALD A. MOELLER
CAPT USAF

Graduate Computer Systems
13 December 1985

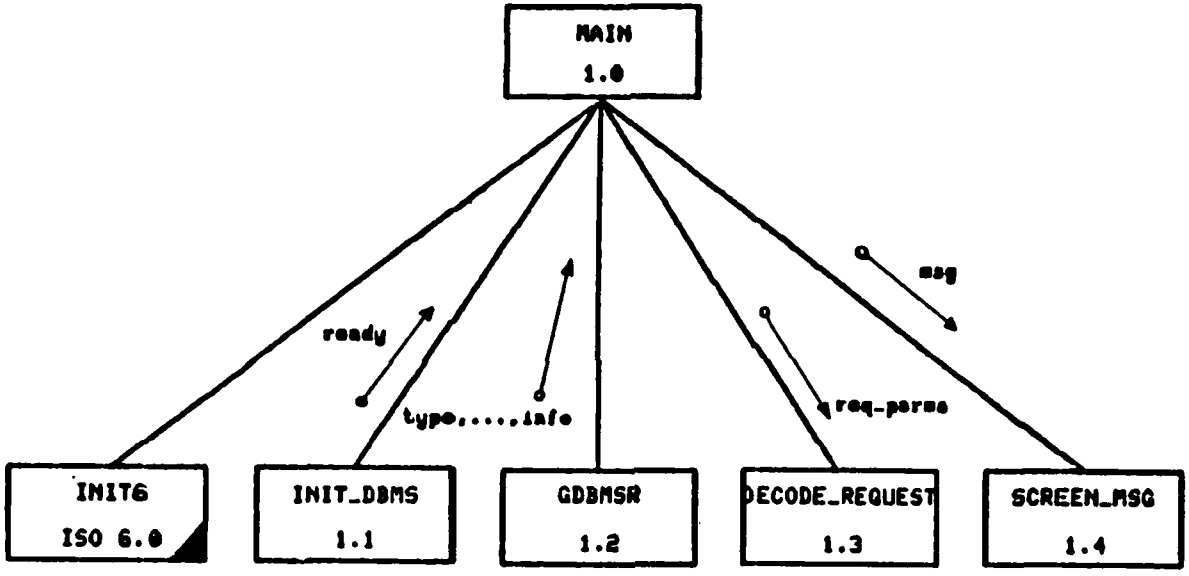


Figure A-1. Computers A and B Main Module.

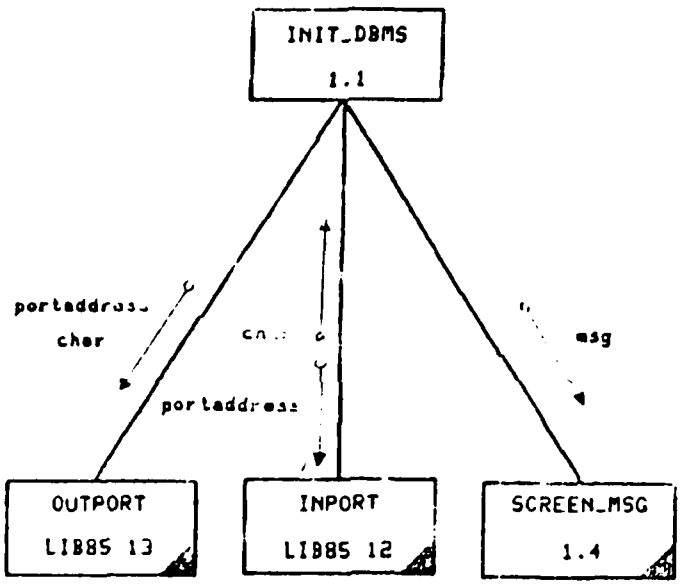


Figure A-2. Computers A and B Init_dbms Module.

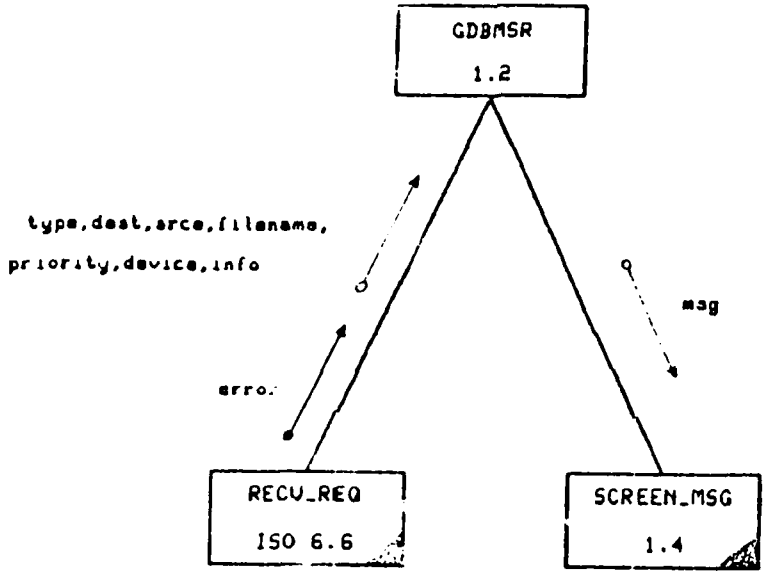


Figure A-3. Computers A and B GDBMSR Module.

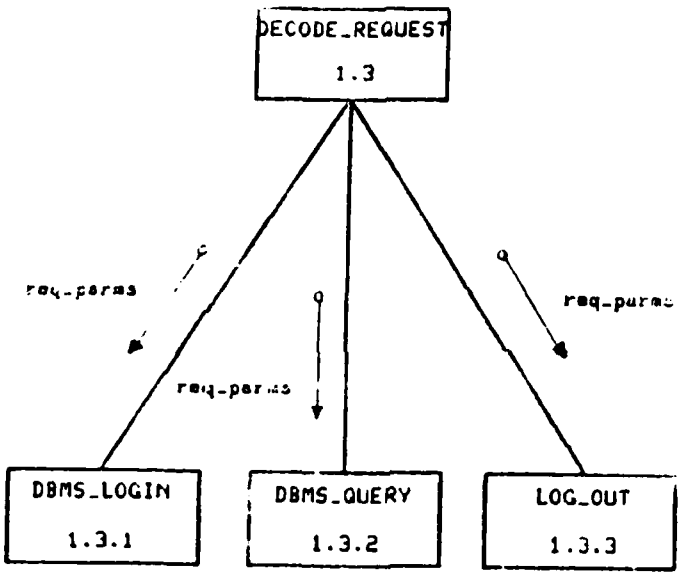


Figure A-4. Computers A and B Decode Module.

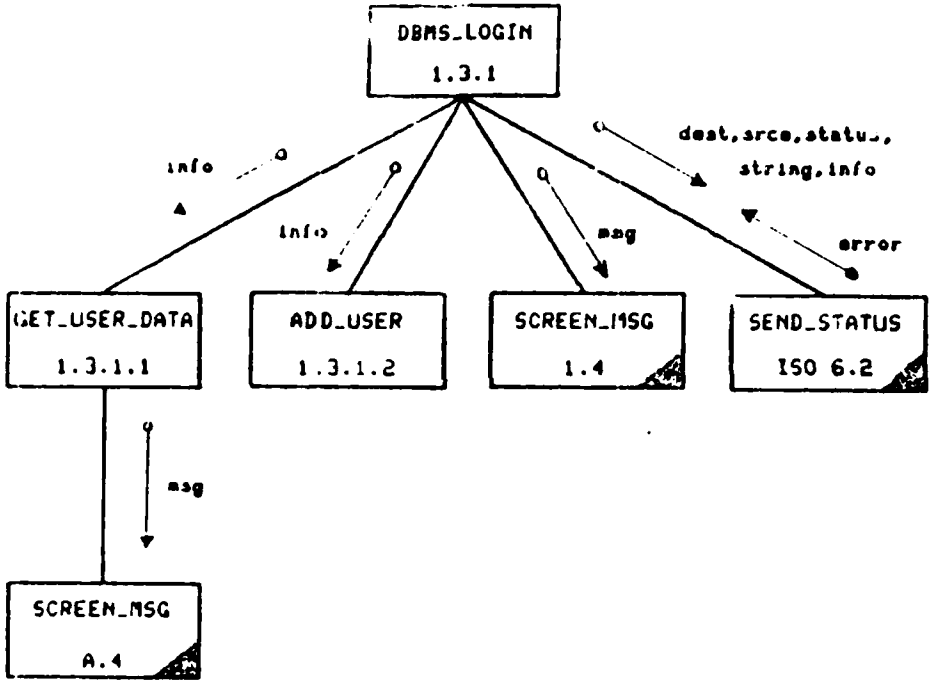


Figure A-5. Computers A and B DBMS_login Module.

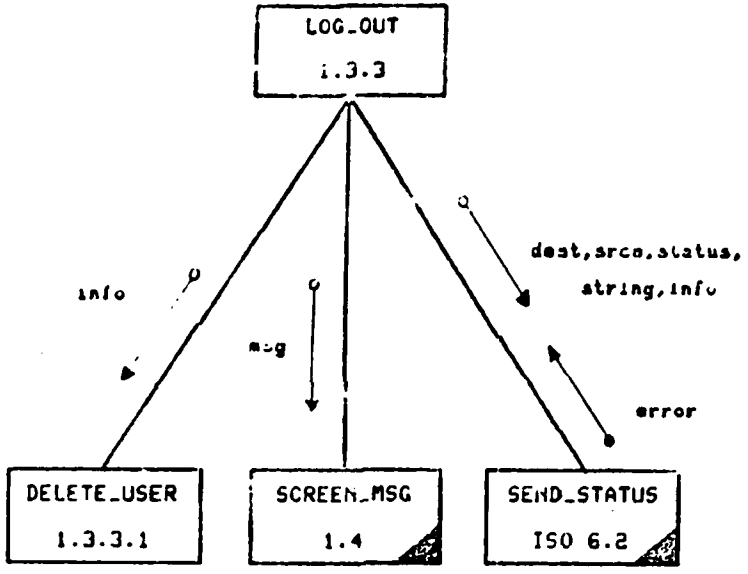


Figure A-6. Computers A and B Log_out Module.

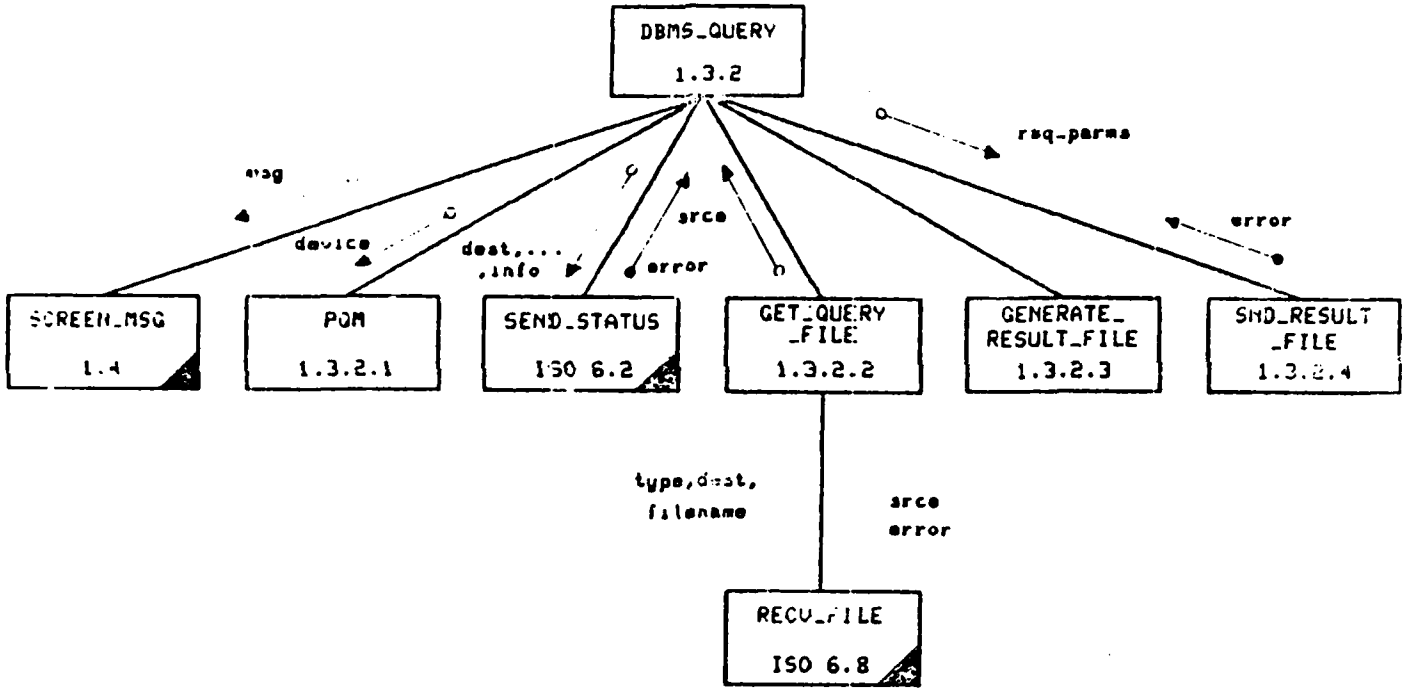


Figura A-7. Computers A and B DBMS_query Module.

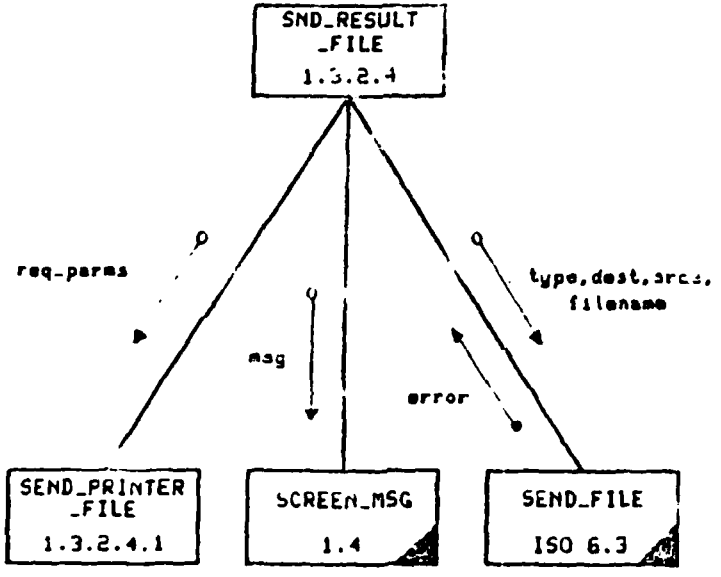


Figure A-8. Computers A and B Snd_results Module.

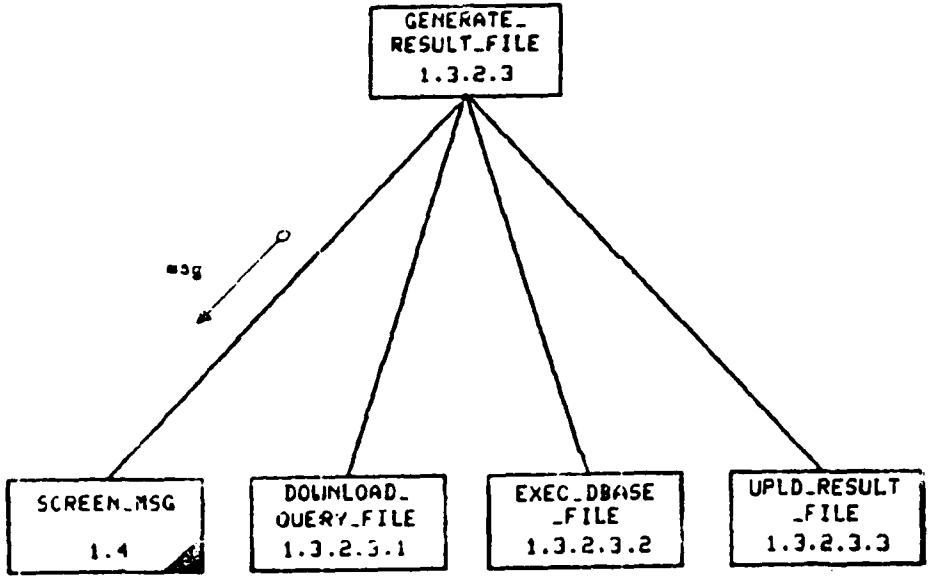


Figure A-8. DBASE Generate_result_file Module.

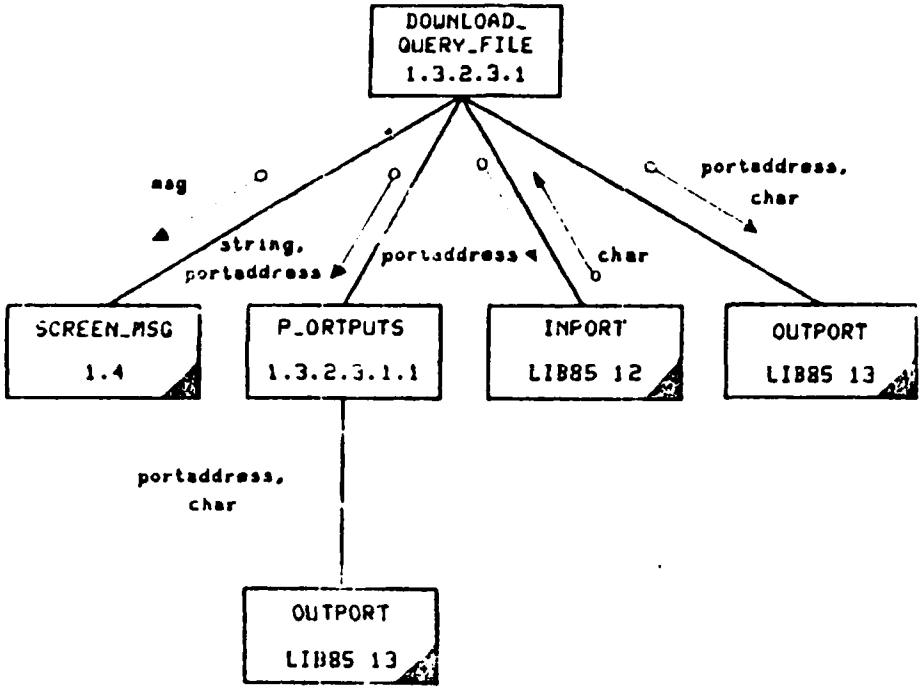


Figure A-10. DBASE Generate_results Module.

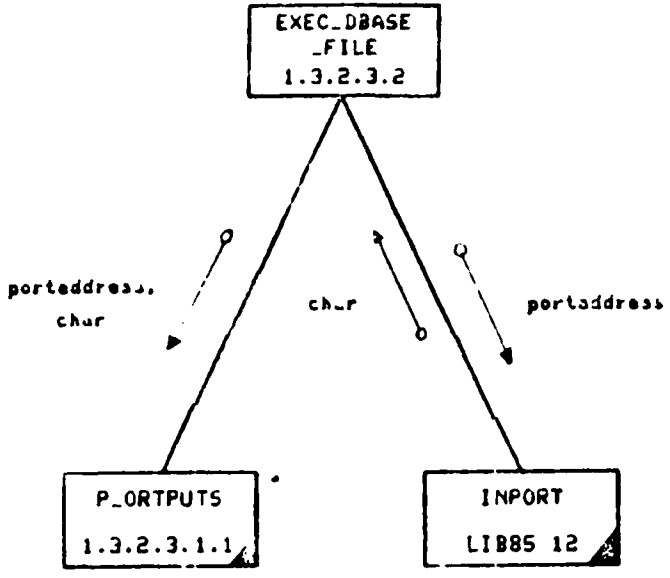


Figure A-11. DBASE Exec_dbase Module.

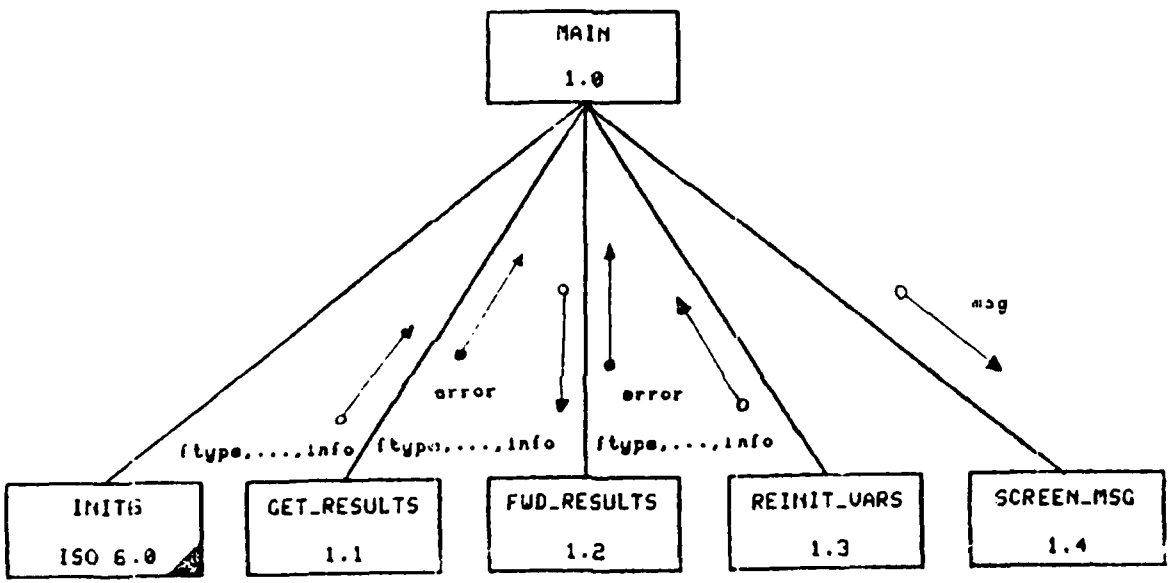


Figure A-13. IFS Main Module.

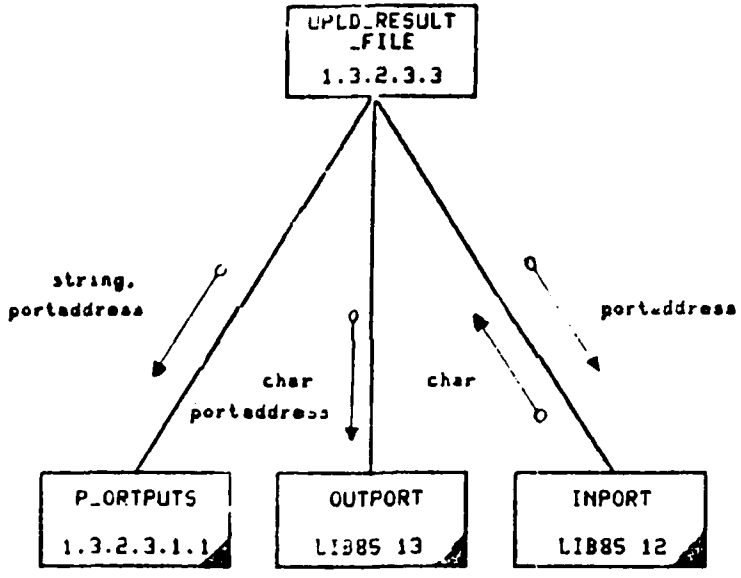


Figure A-12. DBASE Upld_result Module.

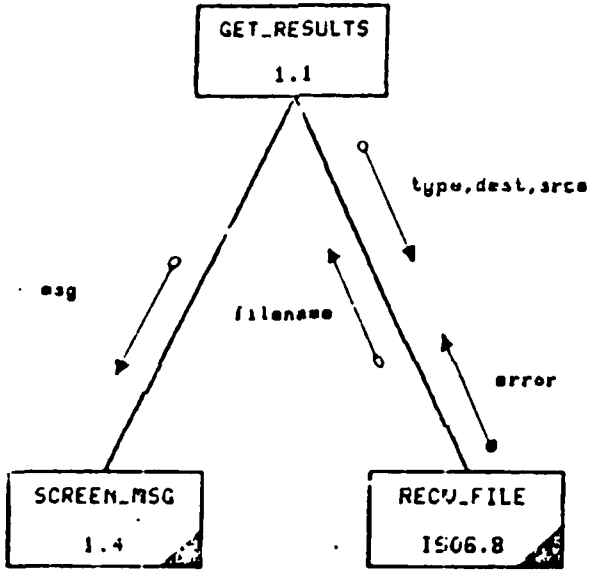


Figure A-14. IFS Get_results Module.

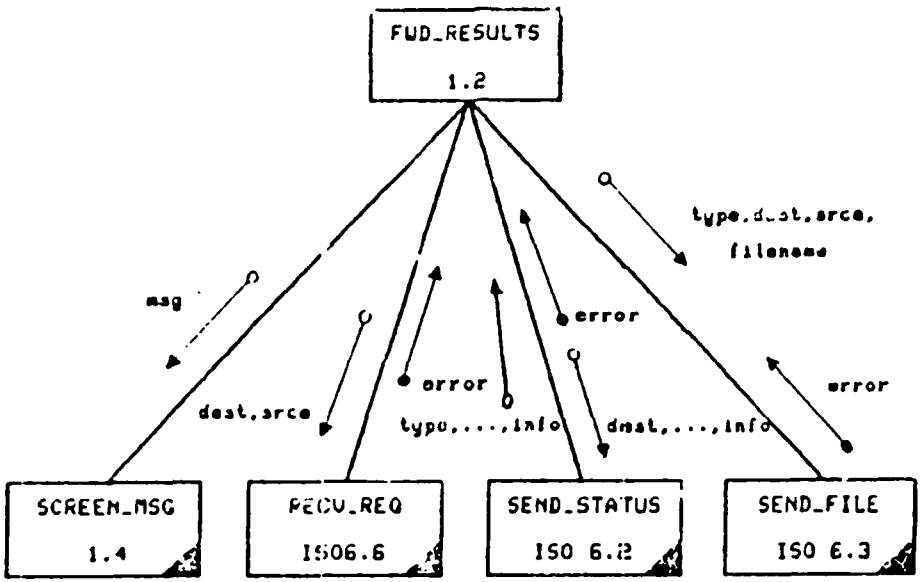


Figure A-15. IFS Fud_results Module.

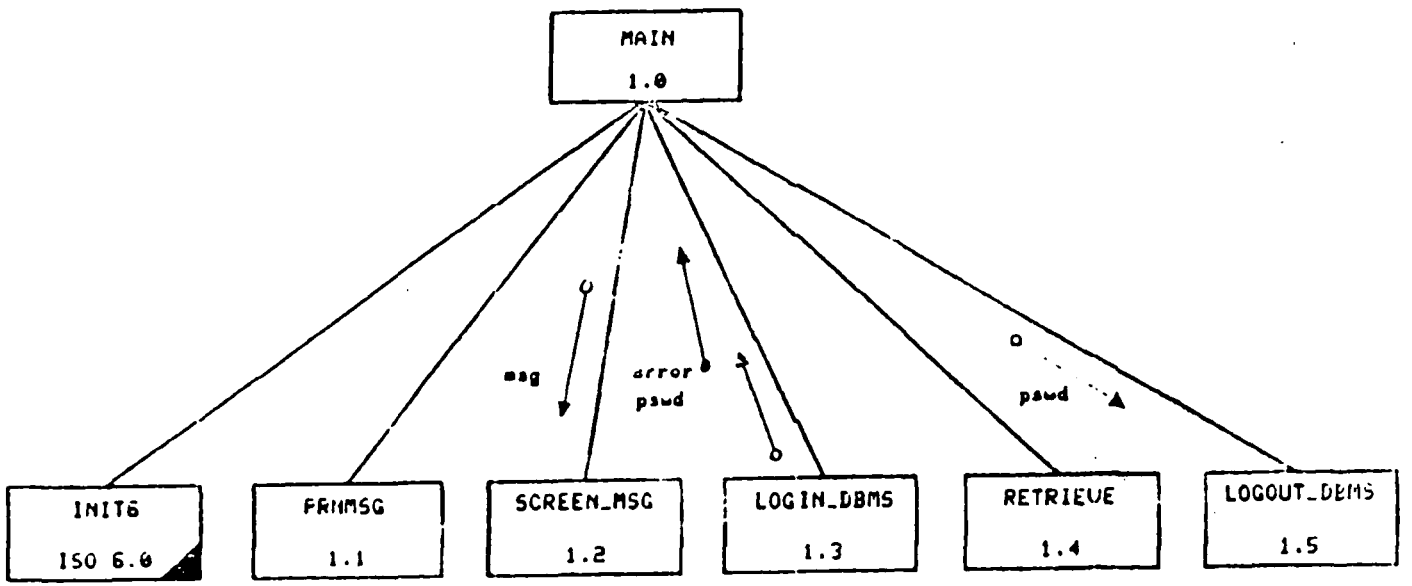


Figure 4-16. Computer C Main Module.

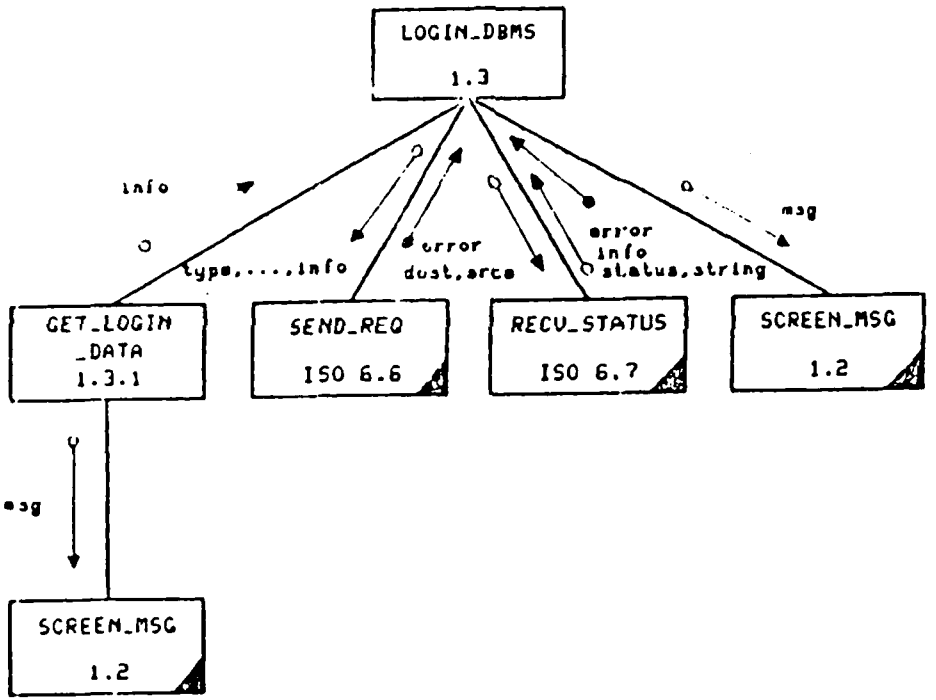


Figure A-17. Computer C Login_dbms Module.

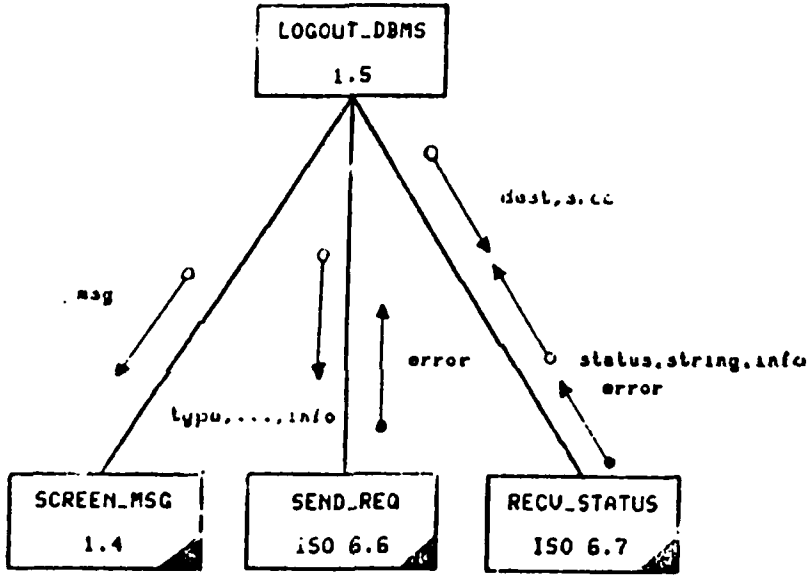
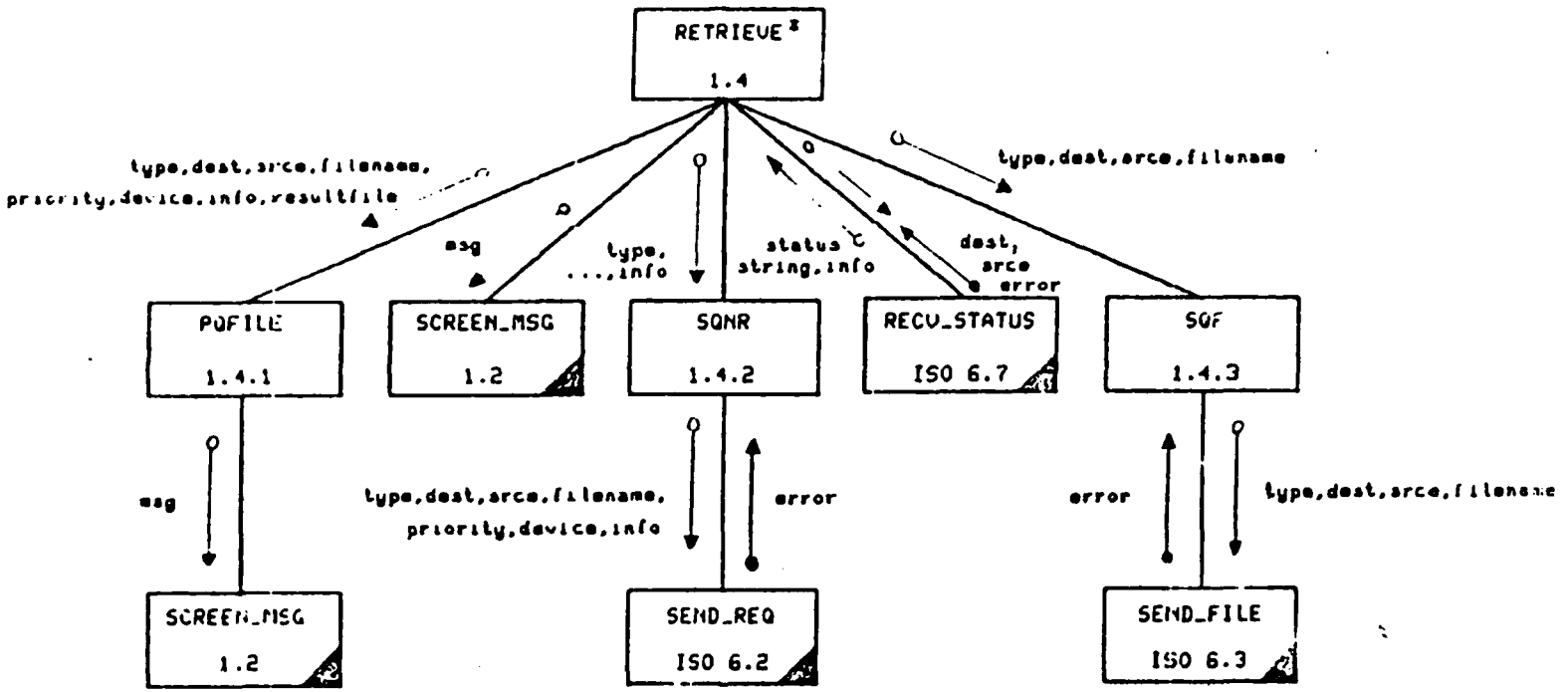
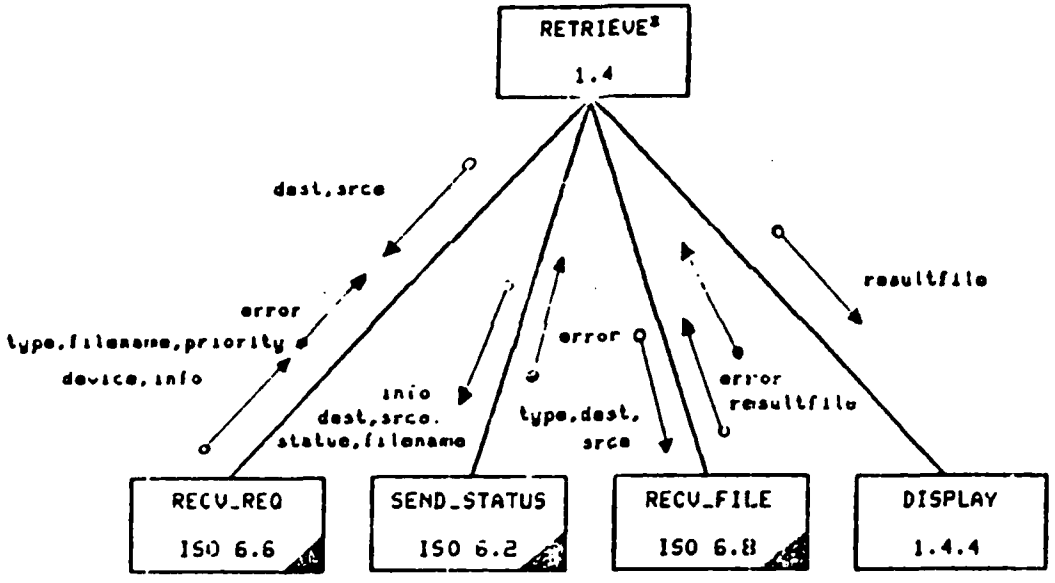


Figure A-18. Computer C Logout_dbms Module.



3 Continued on the following page

Figure A-19. Computer C Retrieve Module.



‡ Continued from the previous page.

Figure A-20. Computer C Retrieve Module.

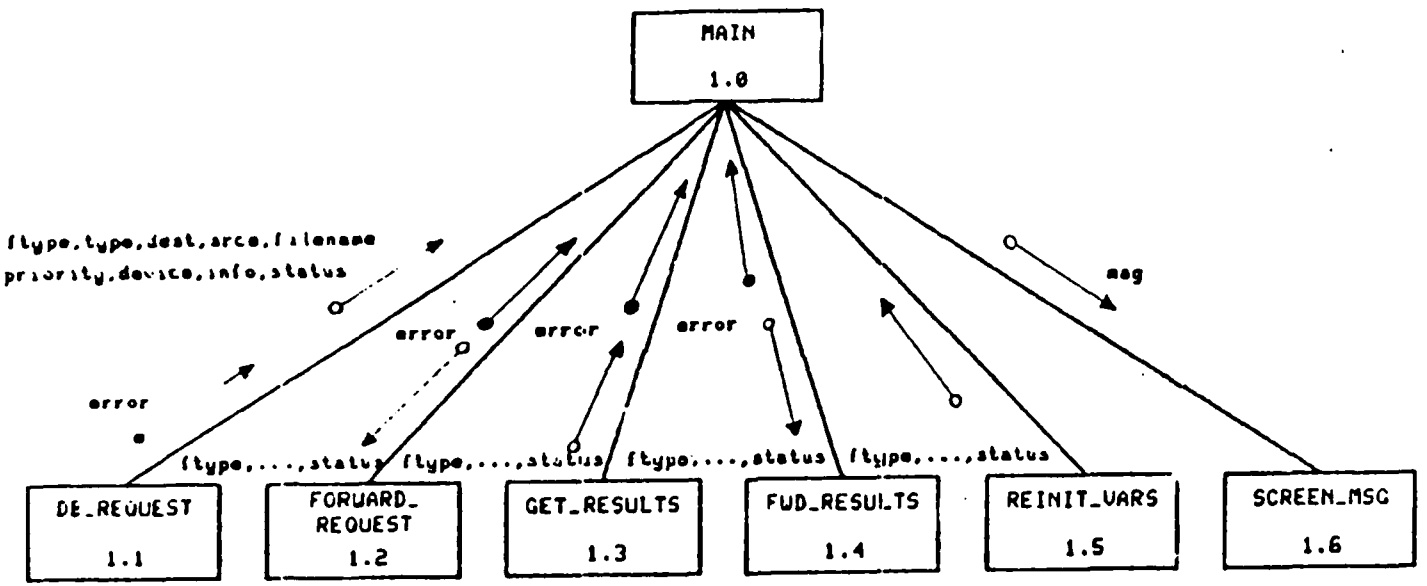


Figure A-21. SFM Main Module.

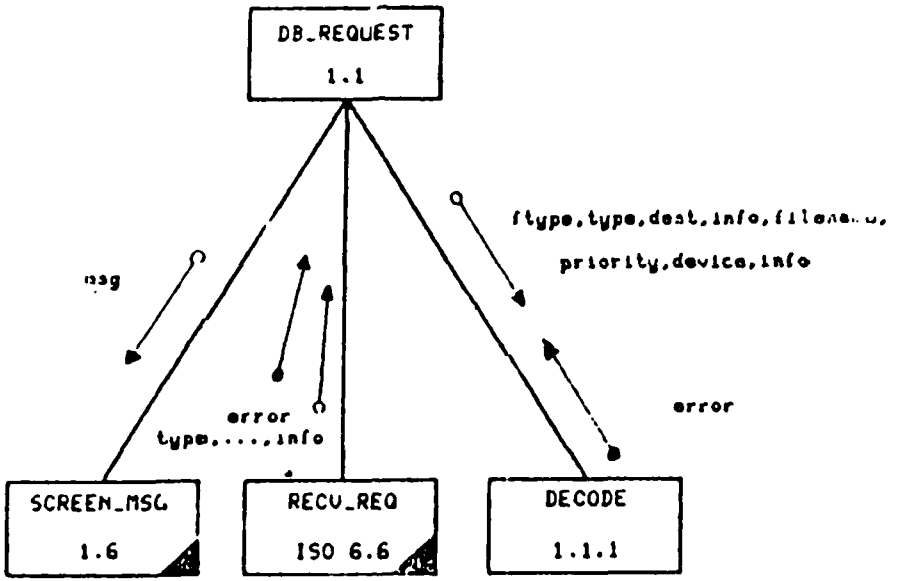


Figure A-22. SFM Db_request Module.

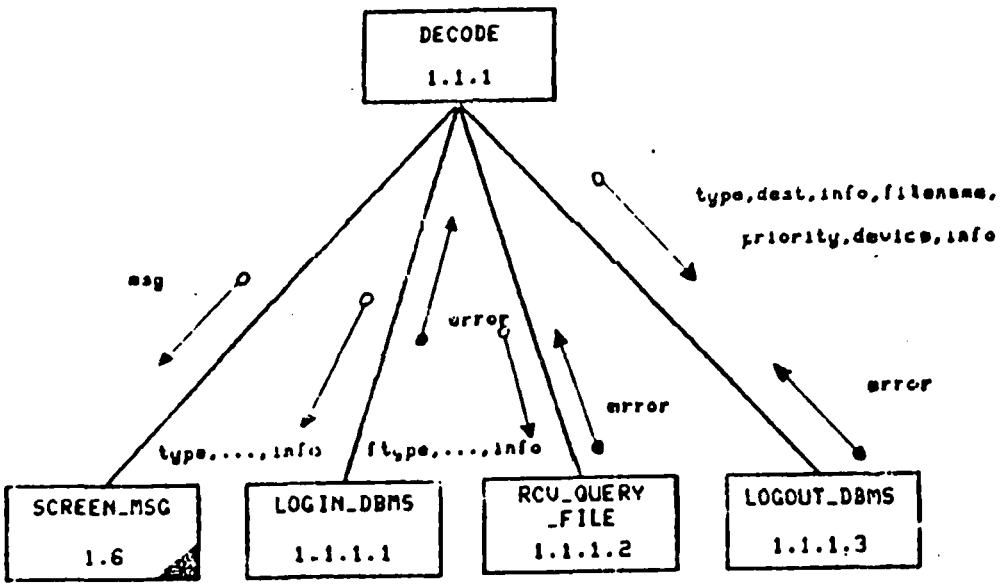


Figure A-23. SFM Decode Module.

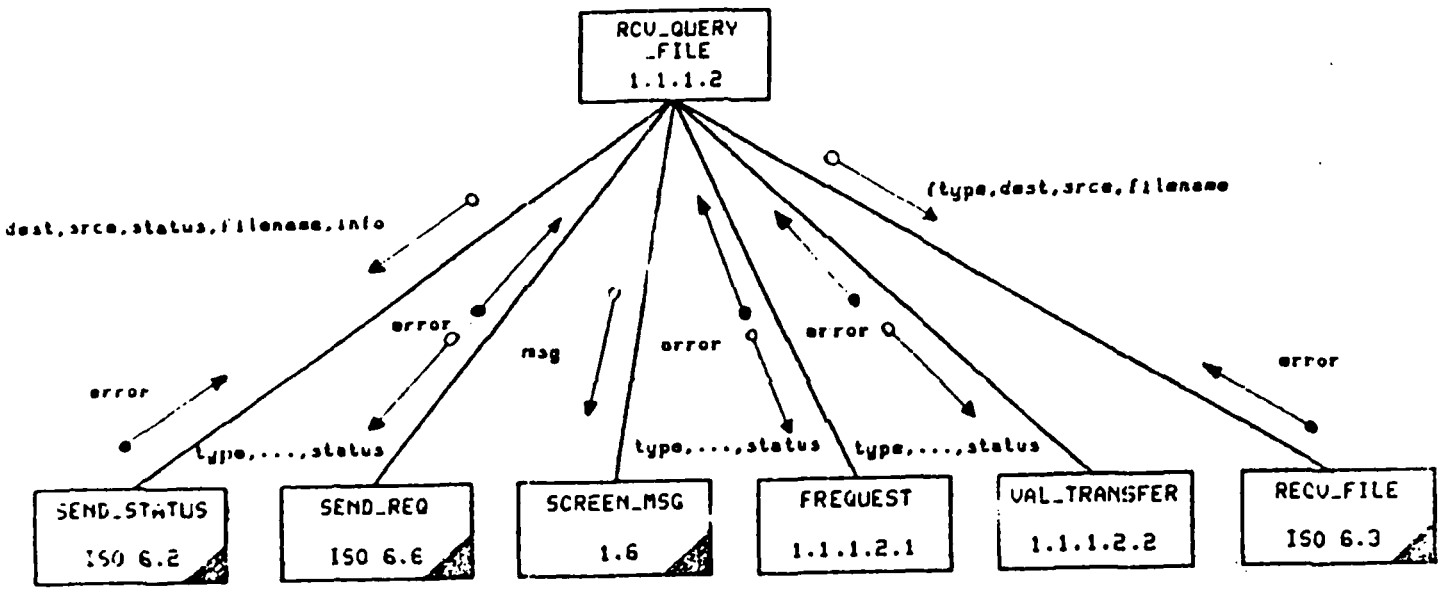


Figure A-2.4. SFM Rcv_query_file Module.

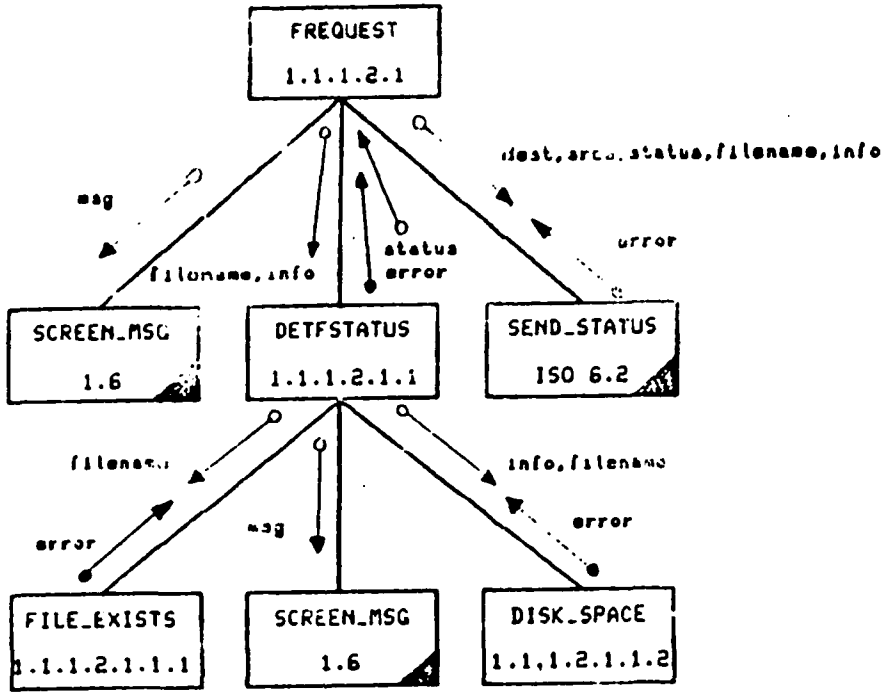


Figure 4-25. SFM Frequest Module.

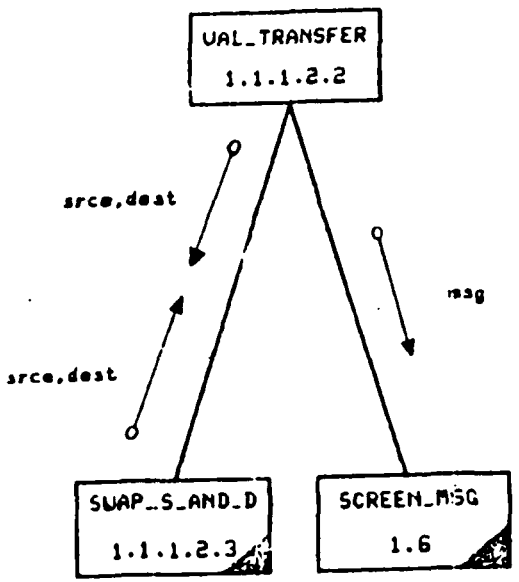


Figure A-26. SFM Val_transfer Module.

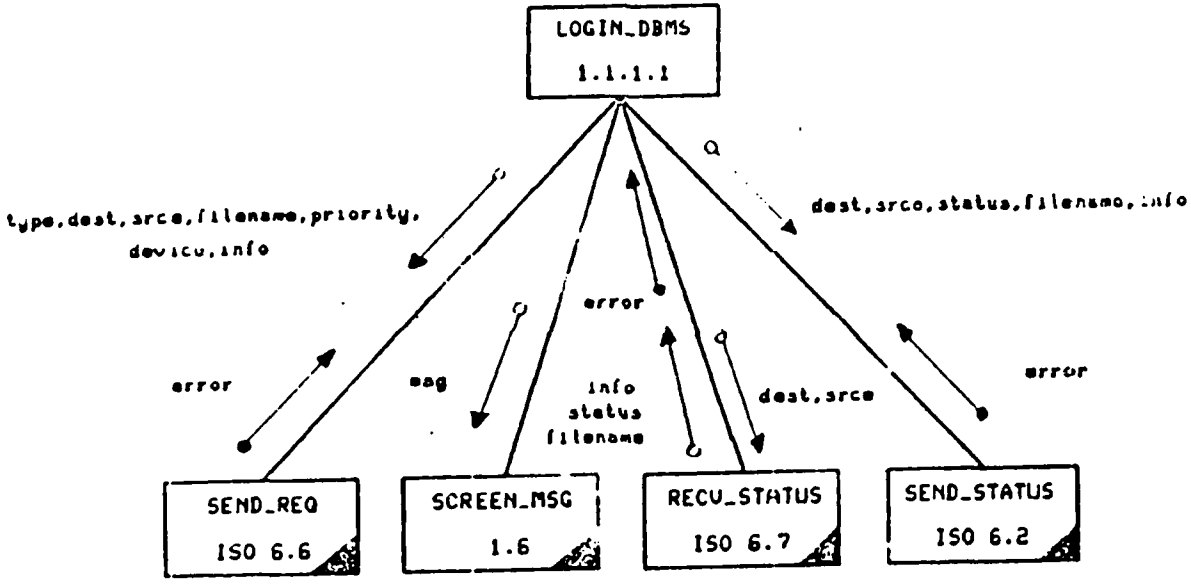


Figure A-27. SFM Login_dbms Module.

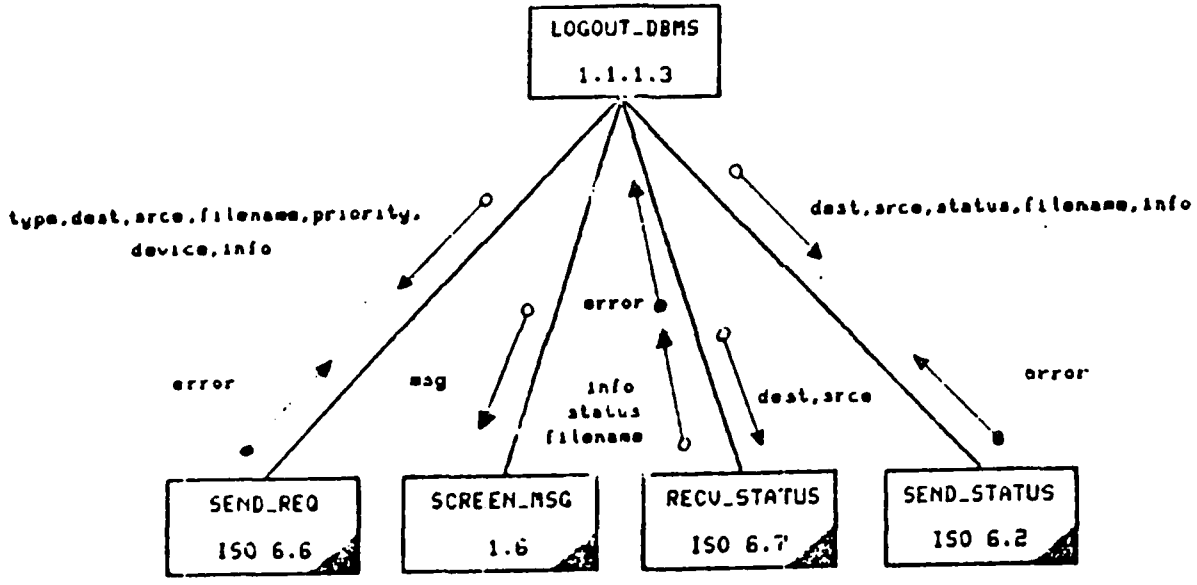


Figure A-28. SFM Login_dbms Module.

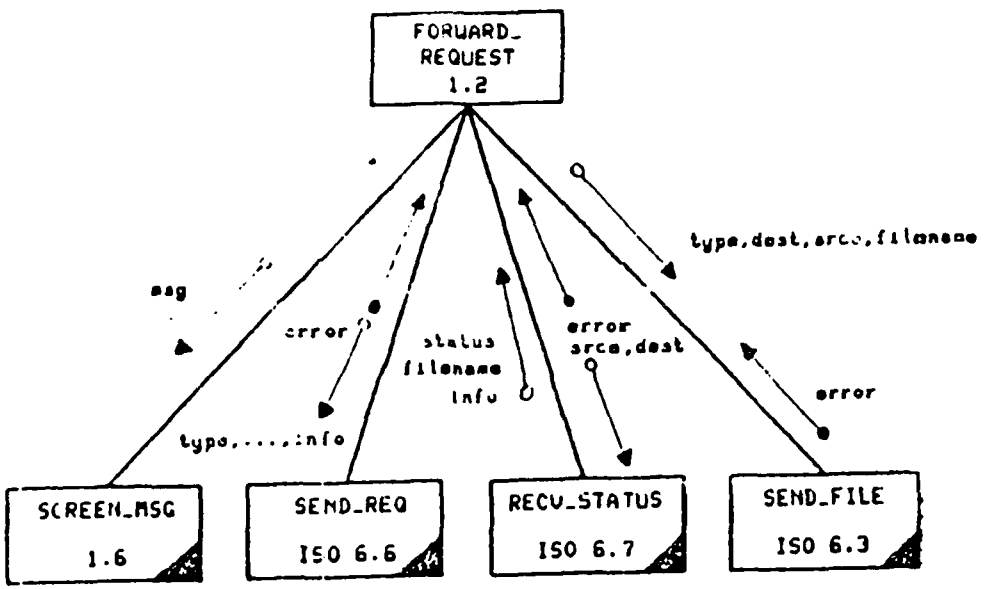


Figure A-29. SFM Forward_request Module.

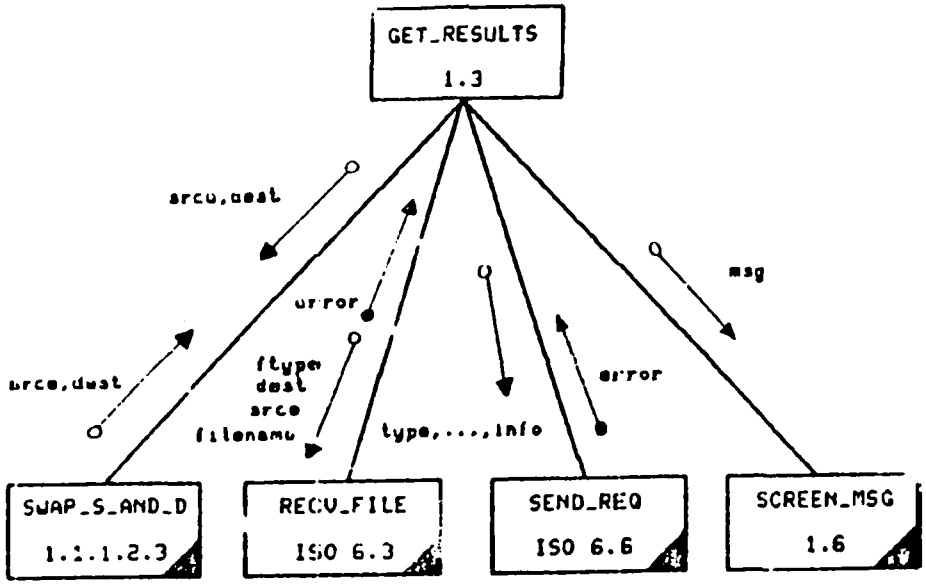


Figure A-30. SFM Get_results Module.

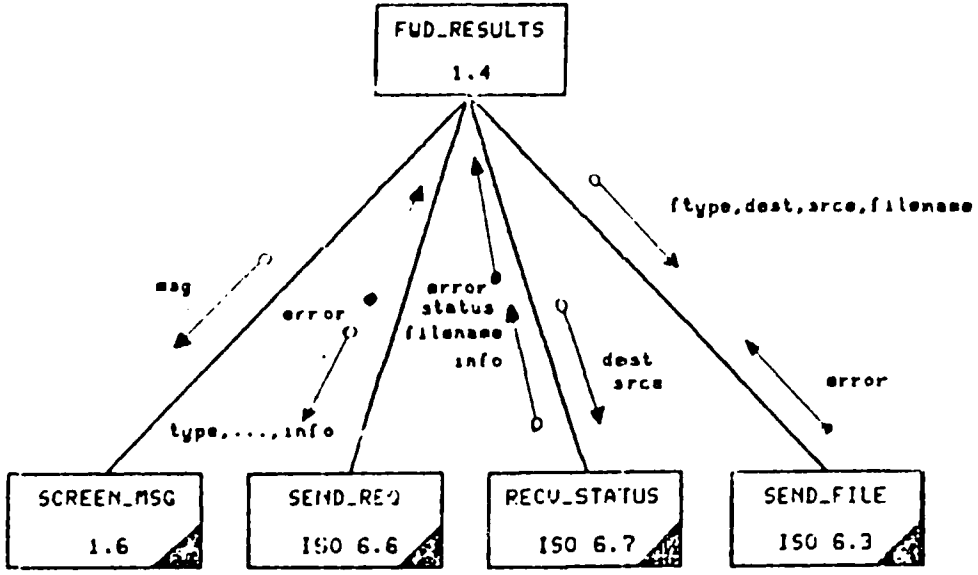


Figure A-31. SFM Fwd_results Module.

The software developed for this thesis is organized into the following six distinct programs: SFM.C, IFS.C, COMPA.C COMPB.C, COMPC.C and DBASE.C. Each of these programs includes a header which details the additional software required for compilation into executable programs on the LSINET located in the LSILAB at the Air Force Institute of Technology. The following steps describe the specific actions required to operate the distributed secure system as implemented for this thesis:

(1) Each of the above mentioned programs, in executable form, must reside on individual 8" floppy disks. All of the disks must be formatted double sided, double density, prior to copying the application software to them. A copy of DBASE.C must reside on the same floppies as both COMPA.C and COMPB.C.

(2) Decide which nodes within the LSINET to locate each of the six programs. The only selections available for COMPA.C and COMPB.C, along with DBASE.C, are System L and System S, since each of these must be connected to a S-100 which is executing dBASE II. The only node not otherwise available for the remaining programs is System B, which must remain as the network's central system. Currently, the SFM and the COMPC programs have been linked with the VTLIB library. If you choose an H29 terminal for these two programs, they will have to be relinked with the H19LIB library. Similarly, the IFS program is currently

configured to run on an H29 terminal rather than a VT100.

(3) Once all the nodes for the distributed secure system have been decided upon, modify the table DDTABL.DAT so that it corresponds to the systems selected. This table consists of four columns, of which the first column is the only one you want to modify. The following three letter identifiers, located in one of the remaining columns of the DDTABL.DAT, should correspond to the systems selected: SFM, IFS, DMA, DMB, DMC.

(4) Copy the updated version of DDTABL.DAT on each of the floppies containing the programs SFM.SAV, IFS.SAV, COMPA.SAV, COMPB.SAV, and COMPC.SAV. This table is required for layer 5 of the network software.

(5) On the S-100 connected to System L, after this system has been cold booted with a copy of dBASE II in drive A and a copy of the database for either computer A{L} or computer B{M} in drive B, enter the following two command lines:

(a) PORTBAUD 9600

(b) STAT CON:=CRT:

These commands are required before the S-100 and System L will talk to one another.

(6) Enter RUN COMPA or RUN COMPB, depending on the computer you chose this node to simulate, on System L after cold booting the system with a floppy containing the current version of the RT-11 operating system in drive A and COMPA.SAV or COMPB.SAV in drive B. The message

"COMMUNICATIONS WITH THE Z-80 SYSTEM NOW ESTABLISHED" should briefly appear, followed by the message "AWAITING REQUEST MESSAGE." This is a non-interactive program, so once these messages appear, the user will not interact with it any further.

(7) Cold boot the S-100 connected to System S with a disk containing dBASE II in drive A and a disk containing the database of either Computer A{L} or Computer B{M} in drive B.

(8) Follow instruction (6) above on System S. Similar messages should appear.

(9) Enter the command line RUN SFM on the system you chose for the secure file manager, with the current version of the RT-11 operating system on the floppy in drive A and the SFM.SAV program on the floppy in drive B. This is not an interactive program, so once the message "WAITING TO RECEIVE A REQUEST MESSAGE" appears, this system is ready to go. (Remember, this program is currently configured to be run on a VT100 terminal.)

(10) Enter the command line RUN IFS on the system you chose for the isolated file store, with the current version of the RT-11 operating system on the floppy in drive A and the program IFS.SAV on the floppy in drive B. This is not an interactive program, so once the message "READY TO RECEIVE A RESULTS FILE" appears, this system is ready to go. (Again, remember that this system is currently configured to be run on an H29 terminal.)

(11) Enter the command line RUN COMPC on the system you chose for computer C{H}, with the current version of the RT-11 operating system on the floppy in drive A and the program COMPC.SAV on the floppy in drive B. This program is the only interactive program available for the entire distributed secure system as implemented.

(12) Once the program is running, select option A, since the user is required to be logged in at both computer A{L} and computer B{M} before either of these databases are queried. To login on both computer A{L} and computer B{M}, ensure that the file PWFILE.DAT is located on both floppies which are in drive B on systems S and L. This file should contain the five letters or less unique identifier you have chosen as your password. Once you have selected option A at computer C{H}, enter the just described password at the program prompt. The message "SENDING LOGIN REQUEST MESSAGE TO COMPUTERS A AND B" should appear. After a couple of minutes or less, the message "USER LOGGED IN OK - PROCEED WITH SESSION" should appear and the user is returned to the original main menu which appeared when the program was first executed.

(13) Select option B and enter the appropriate information when prompted. Always be sure that you enter the drive prefix to any filenames you are prompted for. For instance, when prompted for the name of the query file, enter DX1:TESTA.CMD. Again, remember that this program is configured for a VT100 terminal. If you chose an H29, you

will have to relink the program with the H29LIB library.

(14) After the query has been answered, the user is prompted to determine if he would like to view the results of the query. If so, he enters Y and the response will be presented on the console. Otherwise, he enters N and the main menu reappears.

(15) The user continues to send queries in such a manner until he is finished. At this time, he must logout of the databases. This is accomplished by selecting option C. After a couple of minutes, the message "LOGOUT COMPLETED SUCCESSFULLY" will appear.

(16) To exit from the program, enter option Q at the main menu.

After selecting a query file, the user is prohibited from interacting with the system until the query is either successfully or unsuccessfully completed. Each node in the implemented distributed secure system will flash messages to its associated console indicating the specific action it is currently taking. If the query cannot be answered for any reason, the user on computer C{H} will be notified and the main menu will reappear.

AFIT/GCS/ENG/85D-7

CONFIGURATION GUIDE FOR THE DISTRIBUTED SECURE SYSTEM

APPENDIX C

TO

THE ANALYSIS AND DESIGN

OF A

MULTI-SECURITY LEVEL DISTRIBUTED

DATABASE SYSTEM

RONALD A. MOELLER
CAPT USAF

Graduate Computer Systems
13 December 1985

The following guide provides a detailed listing of the specific modules which must be linked with the programs developed for this thesis to generate executable (.SAV) files:

(1) COMPA.C - Version 1.4, 09/22/85

ISO1C.OBJ, ISO2T.OBJ, ISO2XM.OBJ, ISO2XZ.OBJ, ISO3D.OBJ, ISO4E.OBJ, A5E.OBJ, ISO6E.OBJ, ULIB.OBJ, LIB85.OBJ, CLIB.OBJ, CHDR.OBJ, QLIB.OBJ, RT85.OBJ, H19LIB.OBJ, DBASE.OBJ

(2) COMPB.C - Version 1.4, 09/24/85

ISO1C.OBJ, ISO2T.OBJ, ISO2XM.OBJ, ISO2XZ.OBJ, ISO3D.OBJ, ISO4E.OBJ, B5E.OBJ, ISO6E.OBJ, ULIB.OBJ, LIB85.OBJ, CLIB.OBJ, CHDR.OBJ, QLIB.OBJ, RT85.OBJ, H19LIB.OBJ, DBASE.OBJ

(3) COMPC.C - Version 2.4, 09/15/85

ISO1C.OBJ, ISO2T.OBJ, ISO2XM.OBJ, ISO2XZ.OBJ, ISO3D.OBJ, ISO4E.OBJ, C5E.OBJ, ISO6E.OBJ, ULIB.OBJ, LIB85.OBJ, CLIB.OBJ, CHDR.OBJ, QLIB.OBJ, RT85.OBJ, H19LIB.OBJ, DBASE.OBJ

(4) DBASE.C - Version 1.3, 09/22/85

ISO1C.OBJ, ISO2T.OBJ, ISO2XM.OBJ, ISO2XZ.OBJ, ISO3D.OBJ, ISO4E.OBJ, A5E.OBJ, ISO6E.OBJ, ULIB.OBJ, LIB85.OBJ, CLIB.OBJ, CHDR.OBJ, QLIB.OBJ, RT85.OBJ, H19LIB.OBJ, COMPA.OBJ

(5) SFM.C - Version 1.2, 09/16/85

ISO1C.OBJ, ISO2T.OBJ, ISO2XM.OBJ, ISO2XZ.OBJ, ISO3D.OBJ, ISO4E.OBJ, SFM5E.OBJ, ISO6E.OBJ, ULIB.OBJ, LIB85.OBJ, CLIB.OBJ, CHDR.OBJ, RT85.OBJ, H19LIB.OBJ, DBASE.OBJ QLIB.OBJ

(6) IFS.C - Version 1.0, 09/25/85

ISO1C.OBJ, ISO2T.OBJ, ISO2XM.OBJ, ISO2XZ.OBJ, ISO3D.OBJ, ISO4E.OBJ, IFS5E.OBJ, ISO6E.OBJ, ULIB.OBJ,

CLIB.OBJ, CHDR.OBJ, QLIB.OBJ, RT85.OBJ,
LIB85.OBJ, H19LIB.OBJ, DBASE.OBJ

The modules which are unique to this application other than the six just described above are the following and can found on the DELNET archives disk labeled THESIS.DSS:

(1) A5E.C - Version 3.1, 09/11/85

(2) B5E.C - Version 3.1, 09/21/85

(3) C5E.C - Version 3.1, 09/11/85

(4) SFM5E.C - Version 3.3, 09/30/85

(5) ISO6E.C - Version 3.0, 09/11/85

(6) ISO4E.C - Version 3.1, 09/30/85

AFIT/GCS/ENG/85D-7

PUBLIC'TION ARTICLE FOR

APPENDIX D

TO

THE ANALYSIS AND DESIGN

OF A

MULTI-SECURITY LEVEL DISTRIBUTED

DATABASE SYSTEM

RONALD A. MOELLER
CAPT USAF

Graduate Computer Systems
13 December 1985

Abstract

A multi-security level distributed database system was designed based on database parameters and system requirements provided by Headquarters, Space Division. As the primary step in the analysis of this problem, a thorough investigation into the current state of the art of software verification techniques was made, in order to determine exactly what a computer system's software and hardware could be "trusted" to perform correctly.

Furthermore, a selection was made from available secure local area network alternatives which would yield a solution that would be operational in other than a "system high" mode. The system chosen is currently being researched at the University of Newcastle upon Tyne in Great Britain. This approach involves locating a single security partition on a system which is physically and logically separated from the rest of the network. This separation is performed by a number of software and hardware mechanisms which can be formally proven correct.

Once the distributed secure system design had been suitably tailored for this application, a partial implementation of the design was successfully accomplished upon a local area network being developed at the Air Force Institute of Technology. The test results support the feasibility of this approach to the multi-security level distributed database problem.

Introduction

Over the past twenty years, computer usage has grown to the point that it influences almost every aspect of our lives. Concurrent with this growth has been the development of such system users' needs as the need to share system resources, the need to more effectively utilize system components, and the need for intercomputer communication. This need for the controlled sharing of system resources has grown, not only in the number of people involved, but in the geographic dispersion of both people and resources and their need for increasingly fast access to and transfer of such information. This tremendous expansion has presented new technological problems in many areas, but particularly in that of computer system security.

Recent efforts to build secure computer systems have resulted in limited success. Secure system models have been proposed and several systems have been implemented based on these models. The primary objective of each of these systems is to ensure that a system user has access only to such information for which he is both cleared and has a need to know, an objective which must be proven attainable. However, it is the verification requirements placed on these systems which prevent them from fully achieving their primary goal. Currently available verification techniques simply cannot provide adequate verification assurances for large software programs such as a computer operating system or a database management system. So as the need for inter-

system communication becomes stronger and the number of such systems increase, solutions to the security problems will be sought even if such solutions provide only temporary relief.

Problem Statement

The objective of this study is to propose a database design which will meet the specific requirements of the thesis' sponsor, Headquarters Space Division, Air Force Systems Command. These requirements include the actual database parameters and the operational environment in which the database will reside. In other words, a comprehensive architecture will be defined for a specific database in a particular installation. Furthermore, the study will be limited by the currently available hardware and software or some variation thereof and the methodologies in use today to verify the degree of trustworthiness each system component may assume. Once the design has been accomplished, its implementation will be attempted, using existing Air Force Institute of Technology (AFIT) facilities.

Assumptions

The results of this project will be based substantially on the following assumptions:

1. The physical environment is secure; that is, system operations are performed in physically hardened or guarded facilities.
2. All personnel having authorized access to the system have been cleared through appropriate background investigations to handle classified information.
3. No unauthorized, intentional penetration attempts into the system will be made; in other words, a

nonmalicious environment exists within the local area network.

4. The external communication network's transmission lines are physically secure.

5. The program to implement the recommendations of this study will be well funded but will not have unlimited economic resources.

The above assumptions are based upon the problem specification provided by, and through subsequent conversations with, the thesis sponsor. These assumptions direct the study to specific security related areas, primarily software security, while avoiding other security areas not considered important at this time. Assuming certain system safeguards, particularly a nonmalicious environment, emphasizes the specific direction this project will take and limits the types of potentially compromising situations which may occur. One of the goals of this study is to prohibit situations in which a user of the system unintentionally has access to information for which he is neither cleared nor has a need to know.

Network Requirements

The organization requiring the development of this system, known from this point on simply as the user, has a need for a multi-security level distributed computer system to aid in the planning of aircraft flight paths. This distributed computer system, or local area network, consists of a minimum of three computers, with security levels of the data contained therein ranging from level zero as the most restrictive to level two as the least. The local area

network will be connected to a second network, an external communications network, through which queries and updates to the aircraft flight path database are made. These queries and updates are in the form of messages pertaining to an aircraft's planned flight path and associated weather and surface-to-air missile threats.

Computer A{L}, the first of the three primary hosts, maintains the weather related portion of the database. (For the remainder of this study, whenever one of the principal database computers is mentioned, it will have either a "H", "M", or "L" in braces appended to its single letter identifier. The "H" indicates the computer maintains level zero information, the "M" indicates level one information and the "L" indicates level two information.) The weather data is classified at level two and is received via messages either directly from the external communications network or through an intermediary processor within the local area network. The weather messages describe actual weather conditions within geographical regions by altitude and time. Weather related queries are sent to computer A{L} through the local area network primarily by computer C{H} and involve requests for weather conditions for a specific flight path. Computer A{L} calculates what region(s) the flight path covers and provides computer C{H} all associated weather information.

The second primary host, computer B{M}, maintains the surface-to-air missile (SAM) portion of the database. The

SAM data is classified at level one and is received similarly to the weather data. However, SAM data is received in two different message formats. The first message format provides a SAM site's operational status and contains such information as location, status, and missile type. The second SAM message format describes the characteristics of a particular type of SAM and includes both altitude and range parameters. SAM queries are sent by computer C{H} to computer B{M} and involve requests for the hostile regions a planned flight path covers. Computer B{M} analyzes all legs of the flight path to determine both "free" and "kill" zones. If the flight path covers only "free" zones, computer B{M}'s response is simply "NO DANGER". Otherwise, a "DANGER" message is sent along with the number of SAMs within the zone and all associated site and missile data.

Computer C{H}, perhaps the most important of the three primary hosts, maintains the flight path portion of the database. The flight path data is classified at level zero and is received similarly to both the weather and SAM data above. Once computer C{H} has received a flight path, it queries both computers A{L} and B{M}. Based upon the information received back from the other primary hosts, computer C{H} generates an updated flight path to include the associated weather and SAM data. The updated flight path is then sent either directly to the user via the external communications network or to an intermediary

processor in the local area network.

Database Requirements

The two most important requirements related to the design of the database are the multiple levels of security classifications of the data contained therein and the processing requirements placed on the various hosts of the local area network. The specific security classifications are maintained strictly at the file level; in other words, each record and the fields it consists of are classified according the classification level of the file in which it resides. The processing requirements of each of the primary hosts mentioned above are such that the database must be distributed to provide necessary response times to user queries, as specified by the thesis sponsor. No analysis was made of these response times or host workloads since the thesis problem is structured to be of a distributed nature.

Secure Software Capabilities

In order to more fully understand the fundamental nature of the secure computer system problem, several subjects relating to secure computer systems were investigated, beginning with the most prominent secure operating system design approach, the security kernel. Also investigated was the Bell and LaPadula model, which is the model upon which the security kernel approach is based. The final subject investigated, relating to secure software, was the state of the art of computer software verification methodologies.

As a result of these investigations, it appears that the Department of Defense is confident that the design of secure computer systems using the kernelized approach is the best short-term solution to this complex problem. This opinion is confirmed by many notable computer scientists, including Stanley Ames, when he states that "the security kernel design approach is the most promising methodology currently available that can provide both the internal security and the functional capabilities that many of today's computer systems need (1:22)." Based on both the number of currently existing systems and those in the development process which use the kernel approach, this author would agree with their optimism. However, there are a number of unresolved problems.

One of the most significant problems associated with the kernel design is the confinement problem, which is described as the prevention of a program from leaking sensitive information. The kernelized approach does not address "indirect" information leakage channels, which work through the operating system, by means of response codes and other intended paths (4:284). However, a procedure to detect "indirect" information flow was implemented at the Stanford Research Institute (4:285). A second type of security model, proposed by Denning, specifically addresses the confinement problem and can be certified capable of verifying partial or total confinement of a procedure (6:511).

A second important problem concerns the current state of the specification and verification methodologies. Most of these systems are considered experimental and should not be regarded as a final product (4:280). Again, Mr. Ames even supports this view in his article when he states that solutions for the program correctness problem are still a long way off, particularly for large programs such as an operating system. However, the Ames' article also stated that "even if there was no intent to complete a full mathematical proof, we still have the rigorous review, documentation, and kernel development guidelines that most verification methodologies enforce. These alone will ensure a more secure and reliable system" (1:21). This opinion on Mr. Ames' part raises another, perhaps unanswerable, question, "How secure is secure?"

A final comment of the kernelized approach relates the desired performance objectives with the results observed to date. Naturally, one of the primary objectives of the kernel approach was for the enhanced operating system to minimally degrade system performance relative to its non-kernelized counterpart. Unfortunately, performance has been a serious problem for early attempts at kernelized secure operating systems. In fact, some of these early systems have provided only 10 to 25 percent the performance of similarly configured nonsecure systems. Recent implementations have been more successful at producing systems with "adequate" performance (11:87). Again, the question could

be asked, "How minimal an impact on system performance is truly minimal?"

Secure LAN Alternatives

Having examined computer system security from the standpoint of the mechanisms currently available for a single system, we now turn our attention to those security mechanisms which are in use within a local area network. The specific alternatives analyzed included a trusted network interface, a security controller, and a distributed secure system. This analysis resulted in the selection of the distributed secure system as the system upon which the design of the database described above would be based (see Figure 1). Basically, this system functionally decomposes the network security problem into several smaller problems such that each problem is solvable using software which is within the capabilities of available software verification techniques. Thus, the sum of the individually trustworthy software modules provides the requisite system assurances that accidental disclosures of sensitive information will not occur.

Database Implementation

A partial implementation of the database, as designed using the entity-relationship model, was successfully accomplished. Although the implementation was simplified for this thesis, it provides a strong argument for the validity of the local area network proposed to support the multi-

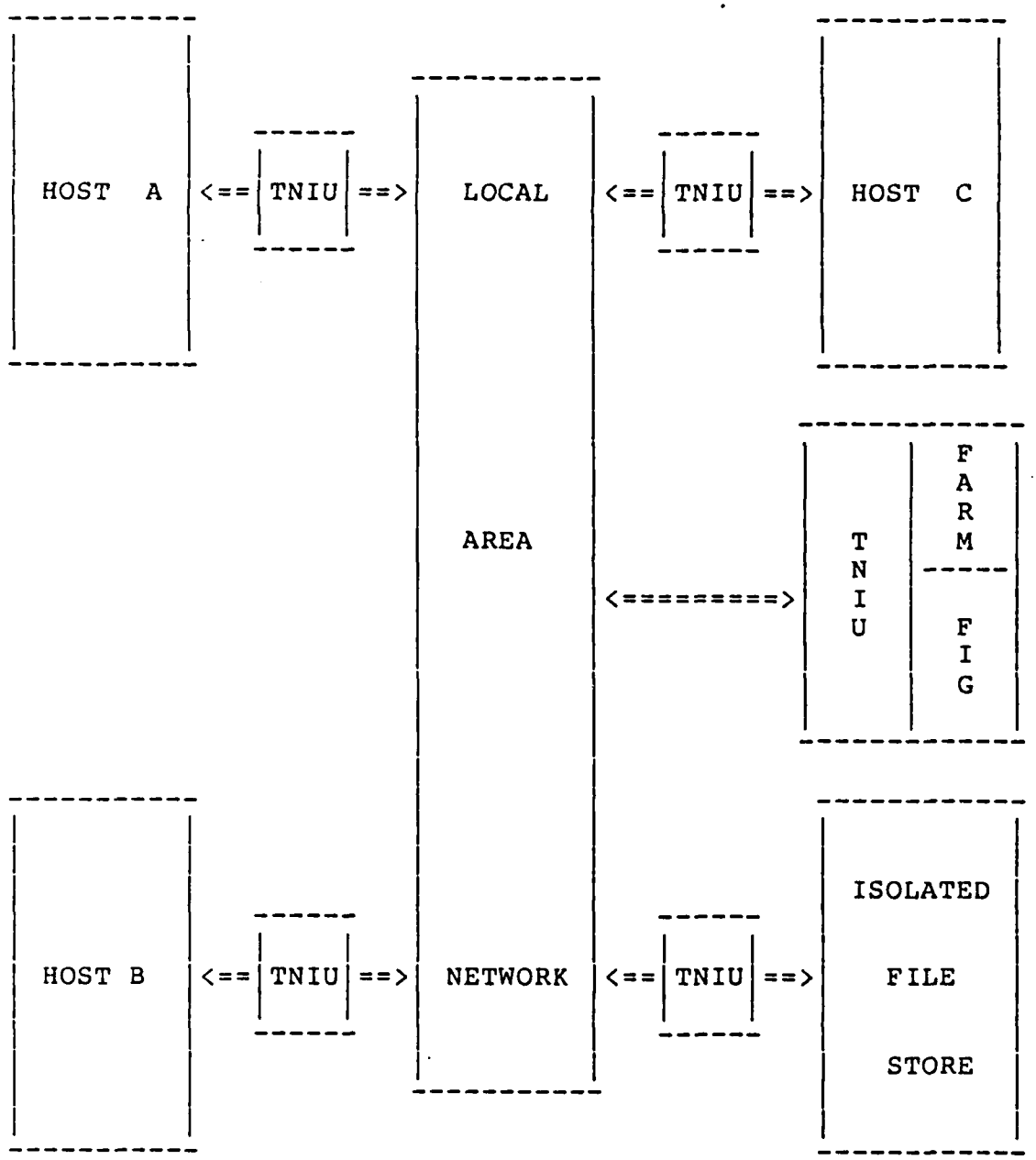


Figure 1. Distributed Secure System

security level database which was provided by Headquarters, Space Division. The specific objectives attained in this phase of the thesis effort were threefold. First, the effects of data encryption on overall database performance were found to be minimal. Secondly, the additional network overhead necessitated by the central position of the secure file manager was found to be substantial but its effects were not so burdensome as to render this design unmanageable. Finally, the total lines of code required for the construction of the secure file manager was found to be within the realm of currently available software verification techniques. However, no attempt was made to prove the correctness of any of the programs written for this project.

Testing and Results

This section describes the tests which were performed on the network (hardware and software) just described. Although these tests are not rigorously defined, their results provide this author's optimism on the verifiability of the solution provided by this thesis to the security problem as described above.

The first objective of determining the impact of the encryption algorithm on system performance was tested by querying the database on computer A{L} with the encryption flag in layer 5 of all programs turned off (see Appendix C for coding details.) In other words, the network was performing normally except that all files were being transmitted in clear text. After executing the command file

three times, the encryption flag in layer five was turned on and computer A{L} was queried three more times.

Table I. Testing Results

	Clear Text		
Start :	21:02:00	21:06:52	21:17:53
Stop :	21:05:08	21:10:03	21:21:02
Time :	3:08	3:11	3:09

	Encrypted		
Start :	21:24:09	21:28:11	21:33:10
Stop :	21:27:20	21:31:20	21:36:23
Time :	3:11	3:09	3:13

The average times for the clear text and encrypted transmissions were 3 minutes, 9.33 seconds and 3 minutes, 11 seconds respectively. With an average time difference of only 1.67 seconds, it is obvious that encrypting the data had little impact on system performance. Moreover, an operational implementation of this design would use an encryption algorithm encoded on an integrated circuit; thus, further improvements in performance would be expected.

The second objective of determining the impact of the additional network message traffic necessitated by the fact that all inter-host communications must be monitored by the secure file manager was tested. Again, using the results of the previous tests of the computer A{L} database, it was observed that it took, on the average, 1 minute, 48 seconds from the time computer A{L} had received notification of a

pending query to the time computer A{L} had completely forwarded the results of the query back to the SFM. As was mentioned before, total execution time from the moment computer C{H} had notified the SFM to the time computer C{H} had completely received the results file was 3 minutes, 11 seconds. Thus, approximately 43% of the total execution time was used for the additional message traffic required by this network design. Although this may seem substantial, the dBASE II command files used for testing, as well as the database itself, were rather simple in nature. Therefore, as the complexities of the queries increase and the database becomes larger, the time for query execution will increase, which will, in turn, reduce the percentage of time spent on network traffic. However, one fact, not previously mentioned, is that all testing was performed on a dedicated network. In other words, there was no other traffic flowing on the LSINET during the testing periods. In a busy network environment, total query execution times would surely be greater than the times recorded during the limited testing performed at this time.

The final objective analyzed during the testing phase of this thesis project involved the size of the secure file manager program which was developed. The total number of lines of source code for this program was approximately 1250 lines. Of these 1250 lines, approximately 500 lines were comments. Given the nature of the development cycle for this project, another 250 lines of code could be removed by

improving the SFM's design. Therefore, even after enhancing the security checking performed within this program, the Secure File Manager would be well under 1000 lines of code. Thus, this program would be of a small enough size to be formally proven using the automated software verification techniques discussed in Chapter 3 (13:95).

Conclusion

The operational implementation of a local area network which supports a multi-security level distributed database is attainable today as evidenced by this thesis' effort at a simplified version of the LAN described above. Although a number of simplifications were included in this first implementation attempt, the attainment of the above mentioned objectives makes this solution worthy of further investigation, particularly in the area of proving software correctness. Hopefully, as the LSINET matures and more computing resources become available, a complete implementation of the LAN will support this author's optimism on the validity of his solution.

Bibliography

1. Ames, Stanley R. et al. "Security Kernel Design and Implementation: An Introduction," Computer, 16: 14-22 (July 1983).

2. Bell, D. E. and L. J. LaPadula. Secure Computer System: Unified Exposition and Multics Interpretation, November 1973 - March 1976. Contract F19628-76-C-0001. Mitre Corporation, Bedford Massachusetts, March 1976 (AD-A023 588).

3. Boeckman, John G. Design and Implementation of the Digital Engineering Laboratory Distributed Database Management System. MS Thesis, ENG 84D-5. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1984.

4. Cheheyl, Maureen H. "Verifying Security," ACM Computer Surveys, 13: 279-339 (September 1981).

5. Date, C. J. An Introduction to Database Systems (Third Edition), Volume I. Reading, Massachusetts: Addison-Wesley Publishing Company, 1982.

6. Denning, Dorthy E. and Peter J. Denning. "Certification of Programs for Secure Information Flow," Communications of the ACM, 20: 504-513 (July 1977).

7. Department of Defense. Department of Defense Trusted Computer System Evaluation Criteria. CSC-STD-001-83. Fort Meade, Maryland: DODCSC, 15 August 1983.

8. Grohn, Michael J. A Model of a Protected Data Management System. Contract F19628-76-C-0025. J.P. Sharp Associates Limited, Ottawa, Canada, June 1976 (AD A035256).

9. Hartrum, Thomas C. Lecture materials distributed in EE 6.90, Software Systems Laboratory. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, September 1984.

10. Landwehr, Carl E. "Formal Models for Computer Security," Computing Surveys, 13: 247-275 (September, 1981).

11. Landwehr, Carl E. "The Best Available Technologies for Computer Security," Computer, 16: 86-95 (July 1983).

12. National Bureau of Standards. Design Alternatives for Computer Network Security. Publication 500-21, Volume 2. Washington: Government Printing Office, 1978.

13. Rushby, John M. and Brian Randell "A Distributed Secure System," Computer, 18: 55-67 (July 1983).

14. Rushby, John M. "The Design and Verification of Secure Systems," Proceedings of the Eighth ACM Symposium on Operating System Principals, 15: 12-21 (December 1981).

15. Shrivastava, S. K. and F. Panzieri "The Design of a Reliable Remote Procedure Call Mechanism," IEEE Transactions on Computers, C-31: 692-697 (July 1982).

16. TM-WD-8513/200/013. Local Area Network Security. Systems Development Corporation, McLean VA, 7 November 1983.

17. Ullman, Jeffrey D. Principles of Database Systems (Second Edition). Rockville, Maryland: Computer Science Press, 1982.

VITA

Captain Ronald A. Moeller was born 20 October 1954 in St. Charles, Illinois. He graduated from high school in Elgin, Illinois in 1972 and attended Elgin Community College, from which he received the degree of Associate in Science in 1974. In February, 1978, he entered Basic Training at Lackland Air Force Base, Texas. In September, 1979, he was accepted into the Airman's Education and Commissioning Program and continued with his undergraduate education at Auburn University, Auburn, Alabama, from which he received a Bachelor of Science in Mathematics in March, 1981. In April, 1981, he attended Officer's Training School at the Lackland Air Force Base Training Annex, Texas, where he received his commission in June, 1981. He attended the Computer Systems Analyst Course at Keesler Air Force Base, Mississippi until September, 1981, when he was reassigned to the 2nd Communications Squadron, Buckley Air National Guard Base, Colorado. He was assigned there until entering the School of Engineering, Air Force Institute of Technology, in May, 1984.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

AD-A172 781

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): AFIT/GCS/ENG/85D-10	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: School of Engineering; 6b. OFFICE SYMBOL (If applicable): AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code): Air Force Institute of Technology, Wright-Patterson AFB, Ohio 45433	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: HQ Space Division (AFSC); 8b. OFFICE SYMBOL (If applicable): SD/DAAX	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code): Lt Col Mekaru, SD/DAAX, Los Angeles, CA 90009-2960	10. SOURCE OF FUNDING NOS. — PROGRAM ELEMENT NO.; PROJECT NO.; TASK NO.; WORK UNIT NO.
11. TITLE (Include Security Classification): See Box 19	
12. PERSONAL AUTHOR(S): Ronald A. Moeller, B.S., Capt, USAF	
13a. TYPE OF REPORT: MS Thesis; 13b. TIME COVERED: FROM _____ TO _____; 14. DATE OF REPORT (Yr., Mo., Day): 1985 December; 15. PAGE COUNT: 183	
16. SUPPLEMENTARY NOTATION	
17. COSATI CODES — FIELD; GROUP; SUB. GR.	18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number): Distributed Database, Multi-Security Levels, Secure Local Area Networks

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

TITLE: ANALYSIS AND DESIGN OF A MULTI-SECURITY LEVEL DISTRIBUTED DATABASE

Thesis Chairman: Dr. Thomas Hartrum, AFIT/ENG

Approved for Public release: IAW AFR 190-1.
9 May 86
LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB OH 45433

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL: Dr. Thomas Hartrum, AFIT/ENG	22b. TELEPHONE NUMBER (Include Area Code): 513-255-2024; 22c. OFFICE SYMBOL: AFIT/ENG

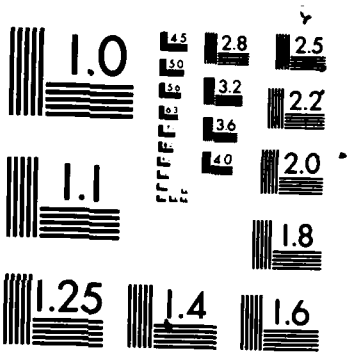
DD FORM 1473, 83 APR — EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

AD-A172 787 ANALYSIS AND DESIGN OF A MULTI-SECURITY LEVEL DISTRIBUTED DATABASE SYSTEM(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING 1/1

UNCLASSIFIED R A MOELLER DEC 85 AFIT/GCS/ENG/85D-10 F/G 9/2 NL



A multi-security level distributed database system was designed based on database parameters and system requirements provided by Headquarters, Space Division. As the primary step in the analysis of this problem, a thorough investigation into the current state of the art of software verification techniques was made, in order to determine exactly what a computer system's software and hardware could be "trusted" to perform correctly.

Furthermore, a selection was made from available secure local area network alternatives which would yield a solution that would be operational in other than a "system high" mode. The system chosen is currently being researched at the University of Newcastle upon Tyne in Great Britain. This approach involves locating a single security partition on a system which is physically and logically separated from the rest of the network. This separation is performed by a number of software and hardware mechanisms which can be formally proven correct.

Once the distributed secure system design had been suitably tailored for this application, a partial implementation of the design was successfully accomplished upon a local area network being developed at the Air Force Institute of Technology. The test results support the feasibility of this approach to the multi-security level distributed database problem.

END

11-86

DTIC