The advent of trusted* computer operating systems

by STEPHEN T. WALKER
Department of Defense
Washington, DC

BACKGROUND

The need to trust a computer system processing sensitive information has existed since the earliest uses of computers. As the effectiveness of computer systems has improved, the desire to utilize them in increasingly more important and consequently more sensitive information processing applications has grown rapidly. Sensitive information must be protected from unauthorized access or modification. But without trusted internal access control mechanisms, the computer has to be treated as a device operating at a single sensitivity level.

Much has been learned about methods of assuring the integrity of information processed on computers since the emergence of operating systems in the early 1960s. Early efforts were primarily concerned with improvements in the effective use of the larger computer centers that were then being established. Information protection was not a major concern since these centers were operated as large isolated data banks. There were many significant hardware and software advances in support of the new operating system demands. Some of these changes were beneficial to the interests of information protection but since protection was not an essential goal at that time, the measures were not applied consistently and significant protection flaws existed in all commercial operating systems.

In the late 1960s, spurred by activities such as the Defense Science Board study (recently reprinted[1]), efforts were initiated to determine how vulnerable computer systems were to penetration. The "Tiger Team" system penetration efforts[2] record of success in penetrating all commercial systems attempted, led to the perception that the integrity of computer systems hardware and software could not be relied upon to protect information from disclosure to other users of the same computer system.

By the early 1970s we had long lists of the ways penetrators used to break into systems. Tools were developed to aid in the systematic detection of critical system flaws. Some were relatively simplistic, relying on the sophistication of the user to discover the flaw,[3] others organized the search into a set of generic conditions which when present often indicated an integrity flaw.[4] Automated algorithms were developed to search for these generic conditions, freeing the "penetrator" from tedious code searches and allowing the detailed analysis of specific potential flaws. These techniques have continued to be developed to considerable sophistication. In addition to their value in searching for flaws in existing software, these algorithms are useful as indicators of conditions to avoid in writing new software if one wishes to avoid the flaws which penetrators most often exploit.

These penetration aids are, however, of limited value in producing trusted software systems. For even if these techniques do not indicate the presence of any flaws it is not possible to prove a positive condition (that a system can be trusted) by the absence of negative indicators (known flaws). There will always be that one remaining flaw that has not yet been discovered.

In the early 1970s the Air Force/Electronics Systems Division (ESD) conducted in-depth analyses of the requirements for trusted systems.[5] The concepts which emerged from their efforts today are the basis for most major trusted computer system developments. The basic concept is a Reference Monitor or Security Kernel which mediates the access of all active system elements (people or programs) referred to as subjects, to all systems elements containing information (tapes, files, etc.) referred to as objects. All of the security relevant decision making functions within a conventional operating system are collected into a small primitive but complete operating system referred to as the Security Kernel. The three essential characteristics of this module are that it be: (1) complete (i.e., that all accesses of all subjects to all objects be checked by the kernel); (2) isolated (i.e., that the code that comprises the kernel be protected from modification or interference by any other software within the system); (3) correct (i.e., that it perform the function for which it was intended and no other function).

Figure 1 is a chronology of some of the major trusted computer system developments that have occurred since 1973. Following the ESD report in 1972, ESD and the MITRE Corporation began a series of efforts to implement security kernel based systems. The early efforts were limited to new operating systems built from scratch.[6] Also in 1973 a design study called the Provably Secure Operating System Study was initiated at SRI International.[7] From this effort emerged a system design specification process which is being widely used in later system developments and the preliminary design for a capability based architecture trusted system. From

* A trusted computer operating system is one which employs sufficient hardware and software integrity measures to allow its use for simultaneously processing multiple levels of classified and/or sensitive information.

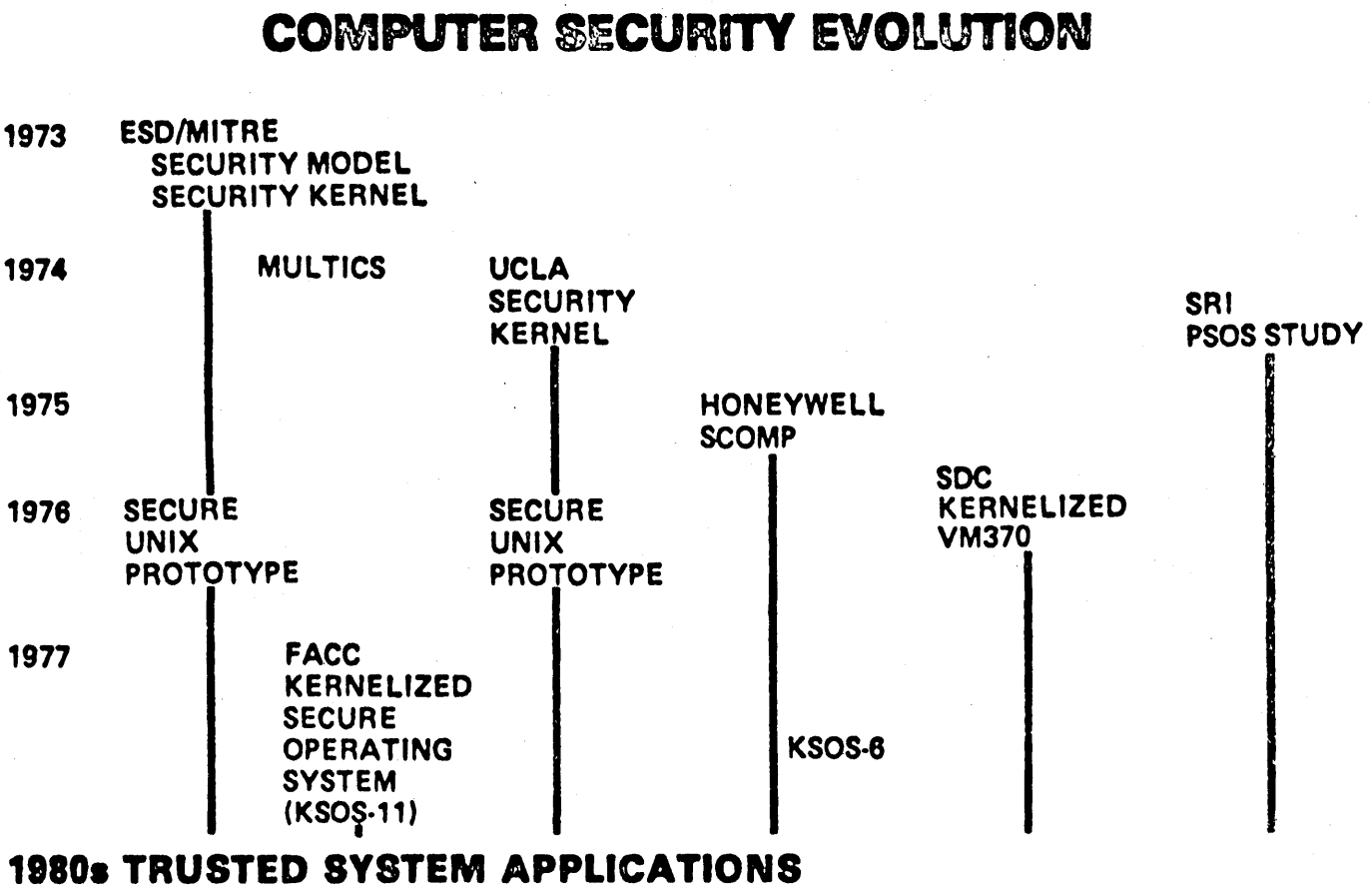

Figure 1.

1974 until 1976 much effort by MIT, Honeywell and MITRE was spent on designs of a security kernel base for MULTICS.[8] In 1974, work began at UCLA to develop a security kernel base for a virtual machine monitor for the DEC PDP-11 computer.[9] From this effort emerged a system design specification process which is being widely used in later system developments and the preliminary design for a capability based architecture trusted system.

In 1975 the Air Force initiated a research effort to design an improved hardware base for use as a Secure Communications Processor (SCOMP). Honeywell won the competitive procurement process with their Level 6 minicomputer. The hardware improvements to the Level 6, designed to improve the efficiency and effectiveness of a security kernel based software system, are called the Security Protection Module and are intended to be compatible options for the standard Level 6 computer.

In 1976 both the MITRE and UCLA projects to implement security kernel based systems began efforts to develop trusted prototypes of the UNIX (TM) operating system.[10,11] Also in 1976, the System Development Corporation (SDC) began the development of a kernelized version of the IBM VM370 operating system (KVM).[12] This system was demonstrated in a preliminary version in October 1979 and is expected to be available for specific DoD applications by late 1980.

In 1977 based on the success of the UCLA and MITRE trusted UNIX prototype developments, an effort was begun to develop a "protection quality" trusted system, entitled the DoD Kernelized Secure Operating System (KSOS) in a two-phase program. In the Design Phase, from August 1977 until April 1978, two competitively selected contractors (Ford Aerospace and Communications Corporation and TRW Inc.) developed detailed system designs. Following a careful evaluation of the two designs, the Implementation Phase contract was awarded to Ford Aerospace in May 1978.[13,14] This phase will implement, by Fall 1980, a production quality trusted operating system which is compatible with UNIX. This effort is sponsored by the Defense Advanced Research Projects Agency and several other DoD agencies, each of which has specific applications in mind for the system.

This KSOS implementation will be on the Digital Equipment Corporation PDP-11/70 computer in order to take maximum advantage of the widespread installed computer base and existing UNIX-compatible applications on that computer. However, the organization of this project has been substantially influenced by the possibility of implementations on hardware other than the PDP-11. The product of the Design Phase was a detailed system level specification.[15] This specification provides a functional description of each module of the security kernel and operating system. This

spec could be used to guide the implementation of versions of KSOS on other hardware architectures. The Honeywell Corporation has undertaken an internally funded KSOS development project for their SCOMP modified Level 6 minicomputer. Other implementations of the KSOS system are being studied by various organizations.

TRUSTED OPERATING SYSTEM FUNDAMENTALS

An operating system is a specialized set of software which provides commonly needed functions for user developed application programs. All operating systems provide a well defined interface to application programs in the form of system calls and parameters. Figure 2 illustrates the relationship between the operating system and application software. The operating system interfaces to the hardware through the basic machine instruction set and to applications software through the system calls which constitute the entry points to the operating system. Applications programs (e.g., A, B and C) utilize these system calls to perform their specific tasks.

A trusted operating system patterned after an existing system is illustrated in Figure 3. The security kernel is a primitive operating system providing all essential security relevant functions including process creating and execution and mediation of primary interrupt and trap responses. Because of the need to prove that the security relevant aspects of the kernel perform correctly, great care is taken to keep the

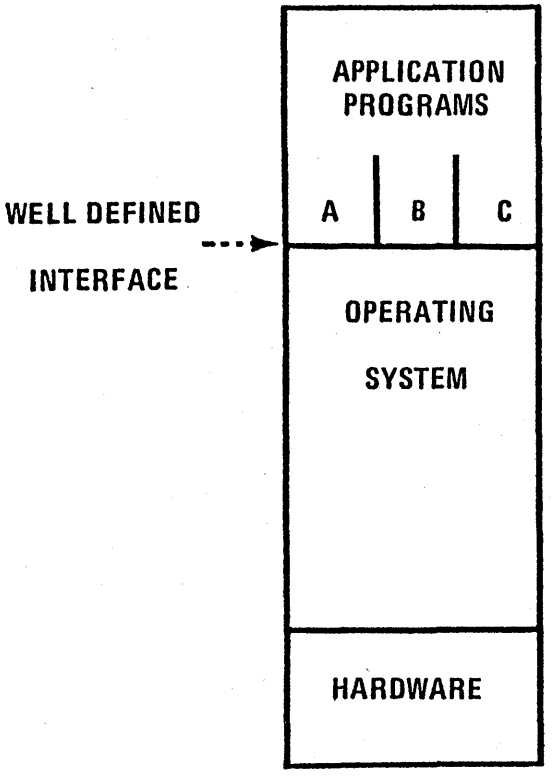

Figure 2.

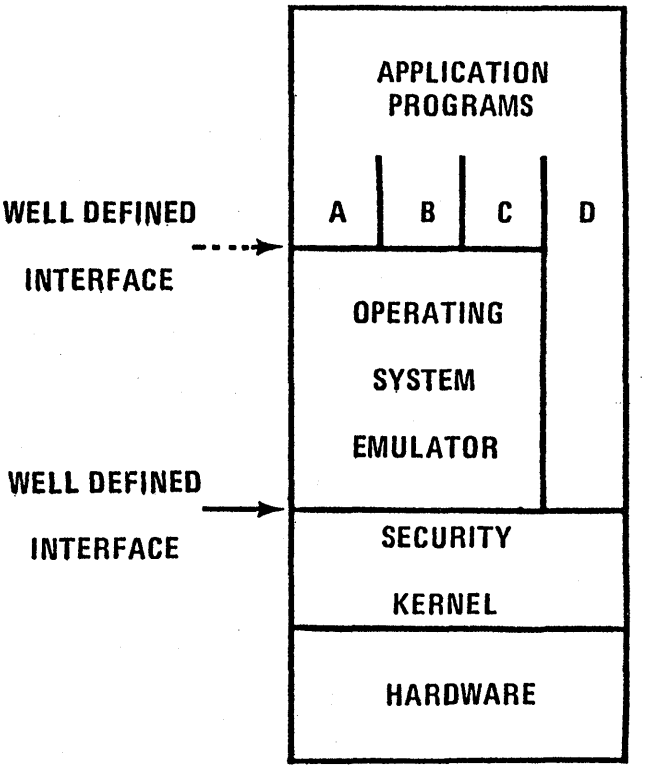

Figure 3.

kernel as small as possible. The kernel interface is a well defined set of calls and interrupt entry points. In order to map these kernel functions into a specific operating system environment, the operating system emulator provides the nonsecurity relevant software interface for user application programs which is compatible with the operating system interface in Figure 2. The level of compatibility determines what existing single security level application programs (e.g., A, B, C) can operate on the trusted system without change.

Dedicated systems often do not need or cannot afford the facilities or environment provided by a general purpose operating system, but they may still be required to provide internal protection. Because the security kernel interface is well defined and provides all the primitive functions needed to implement an operating system it can be called directly by specialized application programs which provide their own environment in a form tailored for efficient execution of the application program. Examples of this type of use are dedicated data base management and message handling systems.

Figure 4 illustrates the relationship between two typical computer systems connected by a network. Each system is composed of an operating system (depicted by the various support modules arrayed around the outside of each box) and application programs (e.g., A, Q, and R in the inner area of the boxes). The dotted path shows how a terminal user on System I might access File X on System II. Working through the terminal handler, the user must first communicate with an application program (A) which will initiate a network connection with the remote computer through the

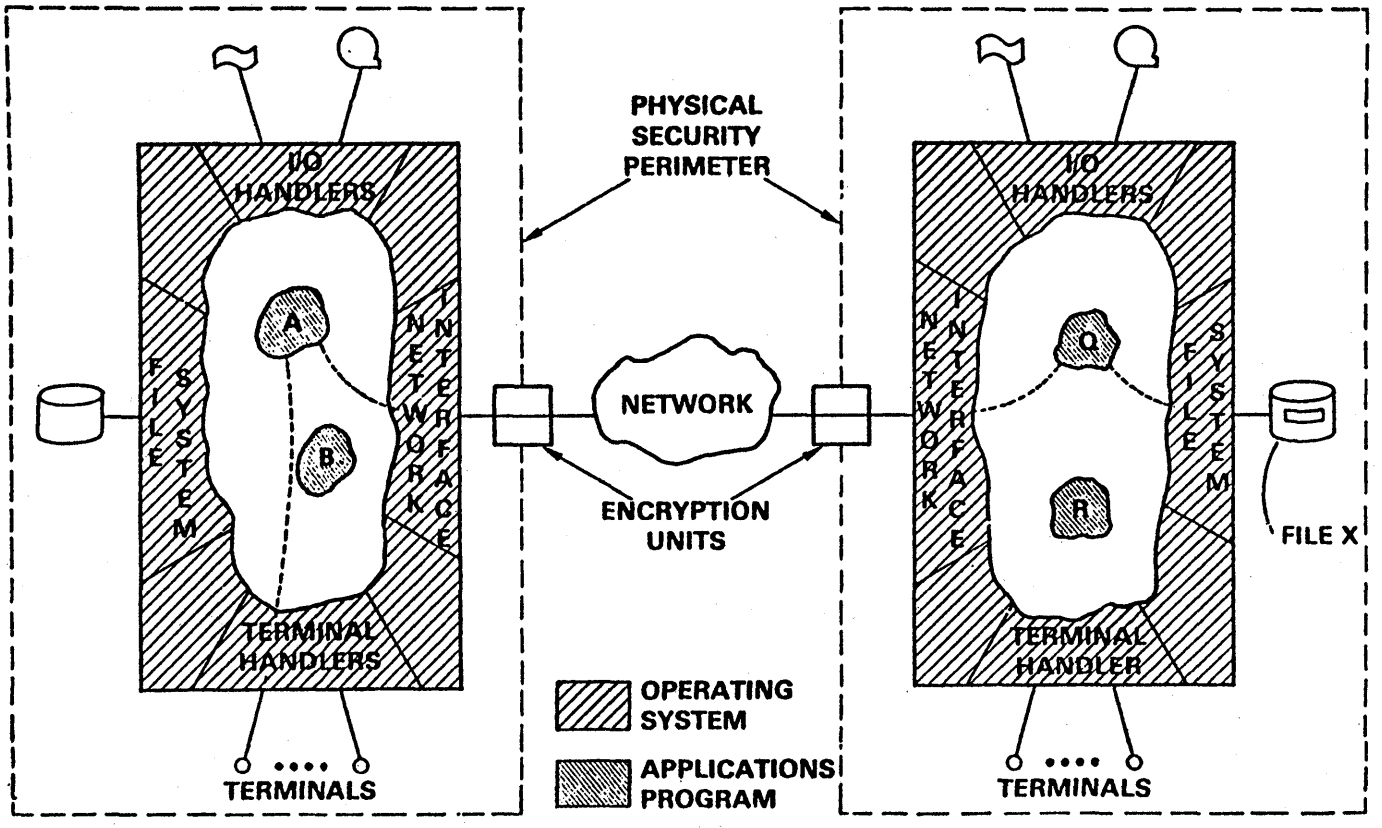

Figure 4.

network interface software. On System II an application program or a system utility (Q) is initiated on the user's behalf to access File X using the file system. Program Q could perform a data base update or retrieval for the user or it could arrange to transfer the file across the network to the local computer for processing.

When this scenario is applied in a secure environment, the two systems are placed in physically secure areas and, if the network is not secure, encryption devices are installed at the secure interface to the network as shown in Figure 4.

Figure 5 illustrates the function of the security kernel in the above scenario. Because the kernel runs directly on the hardware (Figure 3) and processes all interrupts, traps and other system actions, it is logically imposed between all "subjects" and "objects" on the system and can perform access checks on every event affecting the system. It should be noted that depending on the nature of the hardware architecture of the system, the representation of the kernel may have to include the various I/O device handlers. The DEC PDP-11, for example, requires that all device handlers be trusted and included in the kernel since I/O has direct access to memory. The Honeywell Level 6 with the Security Protection Module Option does not require trusted device drivers since I/O access to memory is treated the same way as all other memory accesses and can be controlled by the existing hardware mechanisms.

SYSTEM SECURITY VULNERABILITIES

Protection is always provided in relative quantities. Guaranteed 100 per cent security is not possible with today's physical security measures, nor will it be with new computer security measures. There will always be something which can fail in any security system. The standard approach to achieving reliable security is to apply multiple measures in depth. Traditional locks and fences provide degrees of protection by delaying an intruder until some other protection mechanism such as a roving watchman can discover the attempted intrusion. With computer systems this "delay until detected" approach won't always work. Once an intruder knows about a security flaw in a computer system, he can generally exploit it quickly and repeatedly with minimal risk of detection.

Research on the security kernel approach to building trusted operating systems has produced a positive change in this situation. While absolute security cannot be achieved, the design process for trusted computer systems is such that one can examine the spectrum of remaining vulnerabilities and make reasonable judgments about the threats he expects to encounter and the impact that countermeasures will have on system performance.

A caution must be stated that the techniques described here do not diminish the need for physical and administrative

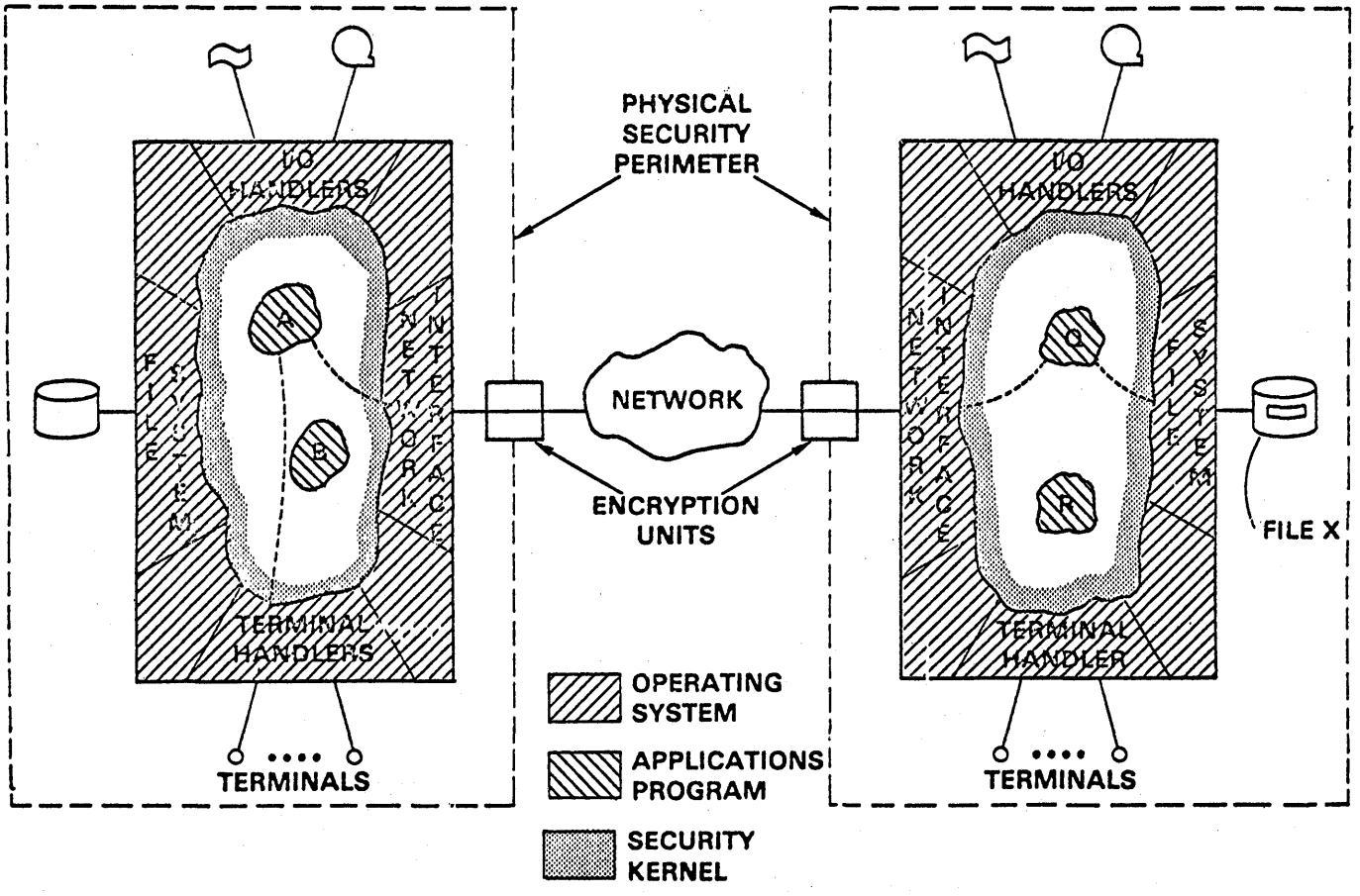

Figure 5.

security measures to protect a system from unauthorized external attack. The computer security/integrity measures described here allow authorized users with varying data access requirements to simultaneously utilize a computer facility. They provide this additional capability which relies upon the existing physical and administrative security measures rather than replacing them.

The nature of traditional physical and administrative security vulnerabilities encountered in the operation of computers with sensitive information is well understood. Only users cleared to the security level of the computer complex are allowed access to the system. With the advent of trusted computer systems allowing simultaneous use of computers by personnel with different security clearances and access requirements, an additional set of security vulnerabilities comes into play. Table I describes one view of this new vulnerability spectrum as a series of concerns. Each of these concerns was not serious in previous systems because there was no need or opportunity to rely on the integrity of the computer hardware and software.

The first category is the Security Policy which the system must enforce in order to assure that users access only authorized data. This policy consists of the rules which the computer will enforce governing the interactions between system users. There are many different policies possible ranging from allowing no one access to anyone else's information to full access to all data on the system. The DoD security policy (Table II) consists of a lattice relationship in which there are classification levels, typically Unclassified through Top Secret, and compartments (or categories) which are often mutually exclusive groupings.[16] With this policy a partial ordering relationship is established in which users with higher personnel security clearance levels can have access to information at lower classification levels provided the users also have a "need to know" the information. The vulnerability concern associated with the security policy is assuming that the policy properly meets the total organizational security requirements.

The second general concern is the System Specification Level. Here the function of each module within the system and its interface to other modules is described in detail. Depending upon the exact approach employed, the system specification level may involve multiple abstract descriptions.[17] The vulnerability here is to be able to assure that each level of the specification enforces the policy previously established.

The next vulnerability concern is the high level language implementation. This category constitutes the actual module

TABLE I—Operating System Security Vulnerabilities

Category	Function	Vulnerability Resolution	Relative Security Risk
Security Policy	Establish security relationship between all system users, resources (e.g., DoD Security Policy)	Review	Moderate
System Specification	Establish policy relationship for each system module (e.g., Parnas I/O assertions)	For each module establish security assertions which govern activity	High
High Order Language Implementation	Transform System Specification Provisions for each module into (e.g. Fortran, PASCAL, C)	Manual or interactive validation that HOL obeys system spec	High
Machine Language Implementation	Transform HOL implementation into binary codes which are executed by hardware	Compiler Testing	Moderate
↑Software (Installation Independent)			
↓Hardware (Installation Dependent)			
Hardware Instruction Modules	Perform machine instructions (e.g., ADD instruction)	Testing, redundant checks of security relevant hardware.	Low -- except for security related hardware
Circuit Electronics	Perform basic logic functions which comprise instructions (e.g., AND, OR functions)	Maintenance Testing	Low
Device Physics	Perform basic electromagnetic functions which comprise basic logic function. (e.g. electron interaction)	Maintenance Testing	Very Low

implementation represented in a high order language (HOL) such as EUCLID[18] or PASCAL. This vulnerability involves the assurance that the code actually obeys the specifications. The next concern on the vulnerability list is the machine code implementation which includes the actual instructions to be run on the hardware. The step from HOL implementation to machine code is usually performed by a compiler and the concern is to assure that the compiler accurately transforms the HOL implementation into machine language.

The next level of concern is that the hardware modules implementing the basic instructions on the machine perform accurately the functions they represent. Does the ADD instruction perform an ADD operation correctly and nothing else? Finally, the last concerns include the circuit electronics and more fundamental device physics itself. Do these elements accurately perform in the expected manner?

As can be seen by analyzing this vulnerability spectrum, some of the areas of concern are more serious than others. In particular, relatively little concern is given to circuit electronics and device physics since there is considerable confidence that these elements will perform as expected. There is a concern with hardware modules, though in general most nonsecurity relevant hardware failures do not pose a significant vulnerability to the security of the system and will be detected during normal operations of the machine. Those security relevant hardware functions can be subject to frequent software testing to insure (to a high degree) that they are functioning properly. The mapping between HOL and machine code implementation is a serious concern. The compiler could perform improper transformations which would violate the integrity of the system. This mapping can be checked in the future by verification of the compiler (presently beyond the state-of-the-art). Today we must rely on rigorous testing of the compiler.

The selection of the security policy which the system must support requires detailed analysis of the application require-

TABLE II—DoD Security Policy

I. Non discretionary (i.e., levels established by national policy must be enforced)

Compartments

	A	B	C
Top Secret			
Secret			
Confidential			
Unclassified			

Partially Ordered Relationship

Top Secret > Secret > Confidential > Unclassified

Compartments A, B, C are mutually exclusive

Example:

User in Compartment B, level Secret can have access to all information at Secret and below (e.g., Confidential and Unclassified) in that compartment, but no access to information in Compartments A or C.

II. Discretionary, "Need to know" - (i.e., levels established "informally").

ments but is not a particularly complex process and can be readily comprehended so the level of concern is not too high for this category.

The system specification and HOL implementation are the two areas which are of greatest concern both because of the complex nature of these processes and the direct negative impact that an error in either has on the integrity of the system. Considerable research has been done to perfect both the design specification process and methods for assuring its correct HOL implementation[19,20,21,22,23] Much of this research has involved the development of languages and methodologies for achieving a complete and correct implementation.[24,25,26]

As stated earlier this vulnerability spectrum constitutes a set of conditions in which the failure of any element may compromise the integrity of the entire system. In the high integrity systems being implemented today, the highest risk vulnerability areas are receiving the most attention. Consistent with the philosphy of having security measures in depth, it will be necessary to maintain strict physical and administrative security measures to protect against those lower risk vulnerabilities that cannot or have not yet been eliminated by trusted hardware/software measures. This will result in the continued need to have cleared operation and maintenance personnel and to periodically execute security checking programs to detect hardware failures. Over the next few years as we understand better how to handle the high risk vulnerabilities we will be able to concentrate more on the lower risk areas and consequently broaden the classes of applications in which these systems will be suitable.

DIRECT VERSUS INDIRECT LEAKAGE PATHS

Computer system security vulnerabilities constitute paths for passing information to authorized users. These paths can be divided into two classes: direct (or overt) and indirect (or covert) channels.[27,28] Direct paths grant access to information through the direct request of a user. If an unauthorized user asks to read a file and is granted access to it, he has made use of a direct path. The folklore of computer security is filled with case histories of commercial operating systems being "tricked" into giving direct access to unauthorized data. Indirect or covert channels are those paths used to pass information between two user programs with different access rights by modulating some system resource such as a storage allocation. For example, a user program at one access level can manipulate his use of disk storage so that another user program at another level can be passed information through the number of unused disk pages.

Unauthorized direct access information paths can be completely eliminated by the security kernel approach since all objects are labeled with access information and the kernel checks them against the subject's access rights before each access is granted. The user who is interested only in eliminating unauthorized direct data access can achieve "complete" security using these techniques. Many environments in which all users are cleared and only a "need-to-know" requirement exists, can be satisfied by such a system.

Indirect data paths are more difficult to control. Some indirect channels can be easily eliminated, others can never be prevented. (The act of turning off the power to a system can always be used to pass information to users.) Some indirect channels have very high bandwidth (memory to memory speeds), many operate at relatively low bandwidth. Depending upon the sensitivity of the application, certain indirect channel bandwidths can be tolerated. In most cases external measures can be taken to eliminate the utility of an indirect channel to a potential penetrator.

The elimination of indirect data channels often affects the performance of a system. This situation requires that the customer carefully examine the nature of the threat he expects and that he eliminate only those indirect paths which pose a real problem in his application. In a recent analysis, one user determined that indirect path bandwidths of approximately teletype speed are acceptable while paths that operate at line printer speed are unacceptable. The assumption was that the low speed paths could be controlled by external physical measures. With these general requirements to guide the system designer it is possible to build a useful trusted system today.

EARLY TRUSTED OPERATING SYSTEM APPLICATIONS

There are a number of classes of applications for which KSOS (either as a full UNIX compatible operating system or in the stand alone kernel mode) is well suited.[29,30] The first is an application called the Guard (Figure 6) in which two commercial untrusted data management systems, op-

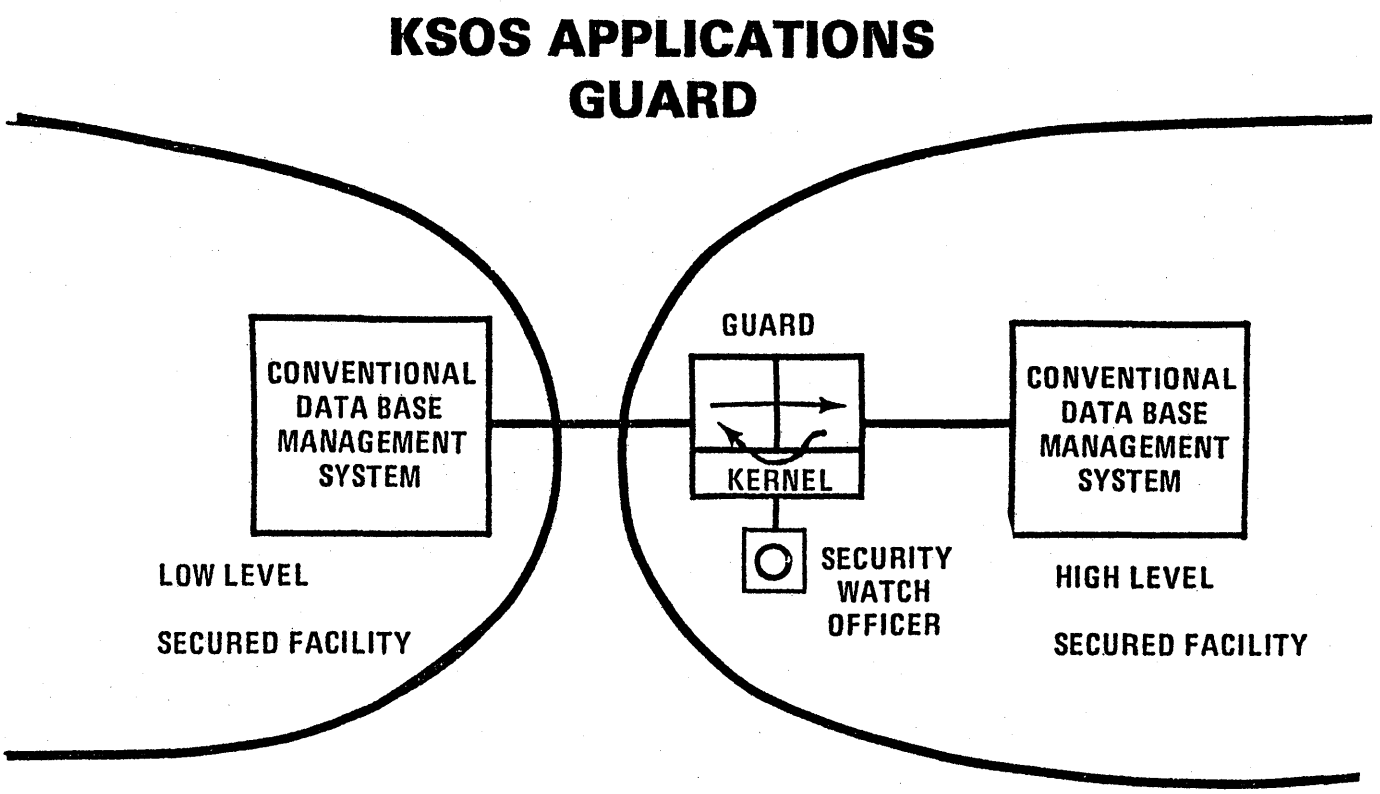

Figure 6.

erating at different security levels, are allowed to interact through a KSOS based security filter. Queries from the low level classified system are passed to the higher system through the Guard. Replies, which might contain information classified at the higher level, are sanitized by either operators or application programs on the Guard. Before information can be passed to the lower level system it must be presented by the security kernel to the Security Watch Officer for a determination of the appropriate classification of the sanitized reply. If the new classification is at or below the clearance level of the lower system, the reply can be forwarded. If not, it will be returned for further sanitization. This relatively simple application provides a very useful function and has wide utility in the DoD.

A second application class is that of trusted network front ends (Figure 7). In the interconnection of DoD computers by sophisticated data communications networks, extensive use is being made of front end minicomputers to offload many of the network protocol and terminal access functions from the mainframe systems. These networks must now be operated in a system high dedicated mode with all computers and terminals operating at the same security level. If these front end systems were implemented on security kernels, subnetworks of computers and terminals, each operating at its own security level, could be established. A set of cooperating trusted network front ends could provide a significant improvement to today's system high operating environment with no change required to the large systems.

A third general class of trusted system applications is that of message handling systems. The DoD has a wide range of requirements for systems of this type. One characteristic which crosses the entire spectrum of this application class and has thusfar not been satisfied is the need for internal integrity within the message system. In many cases security constraints preclude the handling of information from the full set of sources required by an organization because some element of that organization does not have the complete set of clearances required. The result is either a duplication of systems to handle different sources (with the resulting problems of stale or incomplete data) or the inaccessibility of information to sources that require it. If these message handling systems were built on a trusted base such as KSOS, they could make use of the access isolation mechanisms which it provides. Such a system could provide the integration of many information sources into a single environment with sufficient protection to isolate sensitive information.

These are only a few of the applications in the DoD (and in the private sector) that require operating systems with significant levels of internal integrity. Any system that processes sensitive information could benefit from the integrity provided by a trusted operating system. KSOS will not satisfy all the sensitive information handling needs that we now foresee, but as experience in applications like those described here yields a better understanding of more sophisticated applications, the systems which follow KSOS should be able to fulfill the growing number of requirements.

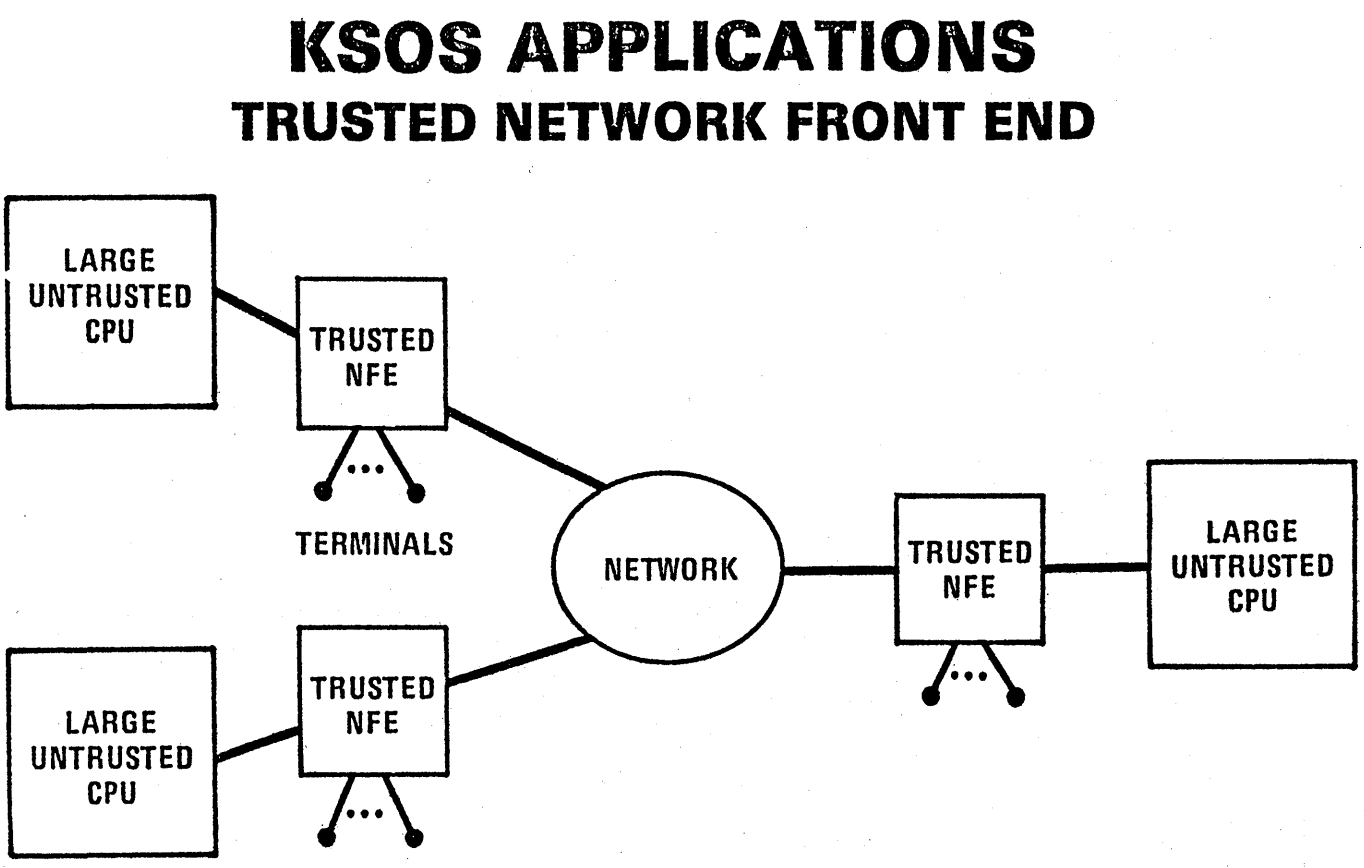

Figure 7.

TRUSTED SYSTEM ACCEPTANCE

The terms "approval," "certification," "accreditation" and "validation" among others have frequently been used to describe some form of acceptance for use of a system in a particular security environment or security mode. Such environments include dedicated mode, periods processing (where the computer is operated at one security classification level for users with the same clearance and "need to know" for a period, stopped, cleared and then run at a different level for a period, stopped, cleared, etc.), system high (where all users are cleared to the same security classification level but may differ in their need to know) and simultaneous multiple security classification levels (referred to as multilevel secure or multilevel security mode where users with different personnel security clearance levels have simultaneous access to the same computer system).

In a multilevel secure environment, where the integrity of the computer hardware and software will be relied upon to the maximum to protect classified information, system approval has been particularly hard to contemplate. However, if a reasonable degree of integrity can be assured for the hardware and software security mechanisms then this combined with appropriate external security provisions should allow acceptance of trusted ADP systems.

The level of integrity afforded by the security kernel mechanism and the formal specification and verification process to which it is subjected, as applied in KSOS and KVM, should be sufficient for approval for use in a number of DoD applications in particular environments. The approval process will involve a detailed analysis of the risks to be encountered by the particular application/environment, the integrity measures inherent in the kernel based hardware and software, and the external physical and administrative measures which can be established.

It is important to understand that security is not a binary decision based only on the characteristics of the operating system or hardware. A particular system installed in one environment may be approved for use, while the same system in a different environment may be unacceptable. With an understanding of the spectrum of vulnerabilities that a trusted system will be subjected to and the external physical and administrative measures that are available, an evaluation of the threat posed by a particular application/environment can be performed to determine the suitability of a particular system in a particular environment. It is possible to establish general categories describing applications and environments and to evaluate systems such as KSOS and KVM to determine in which application/environments they should be suitable. The first applications of KSOS and KVM may be satisfactory for a limited set of environments. Later systems which are able to overcome more of the potential vul-

nerabilities should be acceptable in increasing broad application/environment combinations.

DOD COMPUTER SECURITY INITIATIVE

This paper has thus far described the development and potential use of trusted systems such as the DoD's Kernelized Secure Operating System and Kernelized VM370 System. As significant as these developments are in themselves, there is a much more important "next step" that must be taken. First, some background is needed on the factors which have influenced the actions of the DoD and the computer industry.

The DoD relies on the computer industry to supply its general purpose computer hardware and operating system software. With the exception of limited special purpose systems, most computers in the DoD are commercial products utilizing vendor supplied operating systems. The cost of designing, implementing, and maintaining today's complex systems is so large and the underlying operating system support needs of the DoD are so little different from those of other ADP users that the expense of DoD unique operating systems cannot be justified.

The DoD and others have been for many years asking the vendors to build trusted operating systems. But since we were unable to clearly define what we meant by a trusted system, industry has been reluctant to undertake a serious development because of the high risk that when completed their product might be found unacceptable by either the DoD or other customers. As long as no one was able to demonstrate in detail what was being sought or what constitutes an acceptable product, there was little progress in the development of commercially available trusted systems.

One way to overcome this impasse is for someone (like the DoD) to build a trusted system, demonstrate that is is acceptable in real applications and provide detailed information on the techniques used in the development to the computer industry. If the technology used to build this system is suitable for application in general sensitive information handling environments, then there is a large and rapidly growing marketplace for such a product.

When viewed from this perspective the significance of KSOS and KVM takes on new dimensions. These efforts constitute the existence proof demonstration that a trusted operating system can be built and successfully used in DoD applications. Furthermore the approach used in building KSOS (i.e., the well documented detailed design phase including a top level formal specification of the kernel interface and a formal proof that it enforces an appropriate security policy, followed by an implementation phase) allows immediate transfer of this technology for early use in near term system developments. For the same reasons that the DoD cannot support its own operating system development efforts, it cannot fund multiple vendors to build such systems. But the demonstration of a technology suitable for widespread use in both government and industry should provide sufficient incentive to the computer industry to expend its own development resources to build a suitable line of trusted operating system products.

The DoD wishes to encourage the computer industry to develop, with their own resources, trusted operating systems with security provisions similar to those provided by the KSOS and KVM systems. In support of this, a DoD program is being planned to transfer information concerning the DoD's efforts to develop trusted operating systems and to evaluate industry developed systems which are submitted to the DoD for potential use in sensitive information handling applications.

Two seminars on the DoD Computer Security Initiative have been held in July 1979 and January 1980 at the National Bureau of Standards in Gaithersburg, Md.

SUMMARY

This paper has described the background surrounding the development of several DoD trusted computer systems including the DoD Kernelized Secure Operating System and the Kernelized VM370 System, and the implications of these developments on the future uses of computers. These projects represent general purpose trusted operating systems intended for widespread use. They employ the concepts of a security kernel and an operating system emulator to provide maximum compatibility with existing software applications at a minimum investment of development cost and time.

KSOS and KVM are intended to demonstrate the trusted system development methodology and to provide a base for the useful application of trusted computer systems. They were not intended to be the ultimate answer to everyone's security problems but rather to point the way toward that goal. They also are part of an important initiative by the DoD to transfer an understanding of security kernel technology to industry to assist in the development of commercially available trusted systems.

The work described in this paper is the result of many years of research in trusted computer systems. The technology to build systems described here exists today. There is still much additional research required to develop trusted systems with the full flexibility which will be required in the future. We are recommending continued research into more sophisticated capabilities and we believe that the industry ties established in the DoD Computer Security Initiative will provide strong transfer mechanisms for future research accomplishments.

REFERENCES

1. Ware, Willis H., "Security Controls for Computer Systems, Report of Defense Science Board Task Force on Computer Security," R-609-1, reissued October 1979, Rand Corporation, Santa Monica, CA.
2. Linde, Richard R., "Operating System Penetration," *Proceedings of the 1975 National Computer Conference*, 1975, pp 361-368.
3. Abbott, R. P., et al., "Security Analysis and Enhancements of Laboratory," Livermore, CA, National Bureau of Standards, Washington, DC, NBSIR 76-1041, April 1976.
4. Carlstedt, J., R. Bisbey and G. Popek, "Pattern Directed Protection Evaluation," USC Information Sciences Institute, ISI/75-31, January 1975.
5. Anderson, James P., "Computer Security Technology Planning Study,"

James P. Anderson and Co., Fort Washington, PA, USAF Electronics Systems Division, Hanscom AFB, MA, ESD-TR-73-51, Vols I and II, October 1972 (AD 758206 and AD 772806).
6. Schiller, W. L., "The Design and Specification of a Security Kernel for the PDP-11/45," ESD-TR-75-69, The MITRE Corporation, Bedford, MA, May 1975 (AD A011712).
7. Neumann, P. G., et al., "A Provably Secure Operating System: The System, Its Applications and Proofs," Final Report, Project 4332, SRI International, Menlo Park, CA, February 11, 1977.
8. Schroeder, M., Clark, D., and Saltzer, J., "The MULTICS Kernel Design," *Proceedings of the Sixth Symposium on Operating Systems Principles,* West Lafayette, Indiana, November 1977.
9. Popek, G. and Kline, C., "A Verifiable Protection System," *Proceedings of the International Conference on Reliable Software,* Los Angeles, CA, May 1975.
10. Woodward, J. P. L. and Nibaldi, G. H., "A Kernel-Based Secure UNIX Design," MTR-3499, MITRE Corp, Bedford, MA, November 1977.
11. Popek, Gerald J., et al., "UCLA Secure UNIX," *Proceedings of the 1979 National Computer Conference,* June 1979, pp. 355-364.
12. Gold, B. D., et al., "A Security Retrofit of VM370," *Proceedings of the 1979 National Computer Conference,* June 1979, pp. 335-344.
13. McCauley, E. J. and Drongowski, P. J., "A KSOS - The Design of a Secure Operating System," *Proceedings of the 1979 National Computer Conference,* June 1979, pp. 345-353.
14. Berson, J. A. and Barksdale, Jr., G. L., "KSOS—Development Methodology for a Secure Operating System," *Proceedings of the 1979 National Computer Conference,* June 1979, pp. 365-371.
15. Secure Minicomputer Operating System (KSOS), Computer Program Development Specification (Type B-5), Department of Defense Kernelized Secure Operating System, Ford Aerospace and Communications Corp., WDL-7932, September 1978.
16. Bell, D. E. and LaPadula, L. J., "Secure Computer Systems: Mathematical Foundations and Model," M74-224, The MITRE Corp, Bedford, MA, October 1974.
17. Robinson, L. and Levitt, K. N., "Proof Techniques for Hierarchically Structured Programs," *Communications of the ACM,* Vol. 20, No. 4, April 1977.
18. Lampson B., et al., "Report on the Programming Language EUCLID," *SIGPLAN NOTICES,* Vol. 12, No. 2, February 1977.
19. Popek, G. and Farber, D., "A Model for Verification of Data Security in Operating Systems," *Communications of the ACM,* September 1978.
20. Millen, J., "Security Kernel Validation in Practice," *Communications of the ACM,* Vol. 19, No. 5, May 1976.
21. Feiertag, R. J., et al., "Providing Multilevel Security of a System Design," *Proceedings ACM Sixth Symposium on Operating System Principles,* November 1977.
22. Walker, B., Kemmerer, R., and Popek, G., "Specification and Verification of the UCLA UNIX Security Kernel," Proceedings of ACM SIGOPS Conference, December 1979, to be published in *Communications of the ACM.*
23. Millen, J. K., "Operating System Security Verification," The MITRE Corp., M79-223, September 1979.
24. Roubine, O. and Robinson, L., Special Reference Manual, SRI International, Menlo Park, CA, January 1977.
25. Ambler, A., "Report on the Language GYPSY," University of Texas at Austin, ICSCA-CMP1, August 1976.
26. Holt, R. C., et al., "The EUCLID Language: A Progress Report," *Proceedings of ACM 78 Conference.*
27. Lampson, B., "A Note on the Confinement Problem," *Communications of the ACM,* Vol. 6, No. 10, October 1973.
28. Lipner, S., "A Comment on the Confinement Problem," Fifth Symposium on Operating System Principles, Austin, TX, November 1975.
29. Woodward, J. P. L., "Applications of Multilevel Secure Operating Systems," *Proceedings of the 1979 National Computer Conference,* June 1979, pp 319-328.
30. Padlipsky, M. A., Biba, K. J., and Neely, R. B., "KSOS-Computer Network Applications," *Proceedings of the 1979 National Computer Conference,* June 1979, pp 373-381.