New Thinking About Information Technology Security

Marshall D. Abrams, Ph.D.
Michael V. Joyce

The MITRE Corporation
7525 Colshire Drive
McLean, VA 22102
703-883-6938
abrams@mitre.org

This is the last of three related papers exploring how contemporary computer architecture affects security. It brings together the concepts introduced in the earlier papers and presents a generalized approach to protection, isolation, and access control. We call this approach the *Generalized Trusted Computing Base*. Based upon the "divide and conquer" approach to achieving protection, understandability, and flexibility, the result is a more flexible solution than the rigid hierarchical organization identified in the *Trusted Database Interpretation* or the partitioning introduced in the *Trusted Network Interpretation*.

1 GENERALIZED TRUSTED COMPUTING BASE CONCEPTS

1.1 ISOLATION

The Generalized Trusted Computing Base is based upon structuring a system as a collection of protection domains managed by a separation kernel. As a foundation, a separation kernel provides rigorous isolation, ensuring that the operations within one protection domain do not interfere with another. This quality is fundamental in building a trustworthy environment in which to implement policy-enforcing applications or functions that implement Access Control Policies (Abrams, 1992).

As high-lighted in the second paper in this series, the separation kernel approach has several

This paper was first published in Computers & Security, Vol. 14 No.1 pp. 57-68, © Elsevier Advanced Technology 1995, Oxford, UK; http://www.elsevier.nl/locate/compsec. This is the last of three related papers.

desirable characteristics that make it an attractive structural approach for trusted systems. In the Generalized Trusted Computing Base, the separation kernel specializes in the management of protection domains. This specialization keeps the separation kernel small, easy to understand, and potentially resilient to errors or failures. These characteristics are often cited as desirable traits of "trusted" systems by both designers and evaluators.

The separation kernel provides logical isolation for multiple protection domains on a common bedrock. The same separation exists among multi-processor architectures as an automatic consequence of physical separation. Client-server architectures, for example, support domain separation as a by-product.

Another important characteristic of using a separation kernel is that it does not impose any ordering on the relationships between the protection domains. Since the protection domains are independent of each other, it is not necessary to impose an ordering relationship on the use of objects as resources as was done in the *Trusted Database Interpretation* (DOD, 1985). The principles for composition of protection domains into a Trusted Computing Base, as introduced in the *Trusted Network Interpretation* (DOD, 1985), remain unchanged.

Figure 1 illustrates the three functions included in the separation kernel and the relationship of several protection domains, the separation kernel, and the bedrock. The separation kernel performs the protection domain management operations. The separation kernel obtains resources from the bedrock and transforms those resources into protection domains. At the end of its processing cycles, the separation kernel destroys the protection domains. The separation kernel also contains an interdomain communications function that establishes communications links between protection domains. The links could be static links established during the system initialization or dynamically created links. In the case of dynamically created links, the link is only established after adjudication by the portion of the Reference Validation Mechanism in the separation kernel.

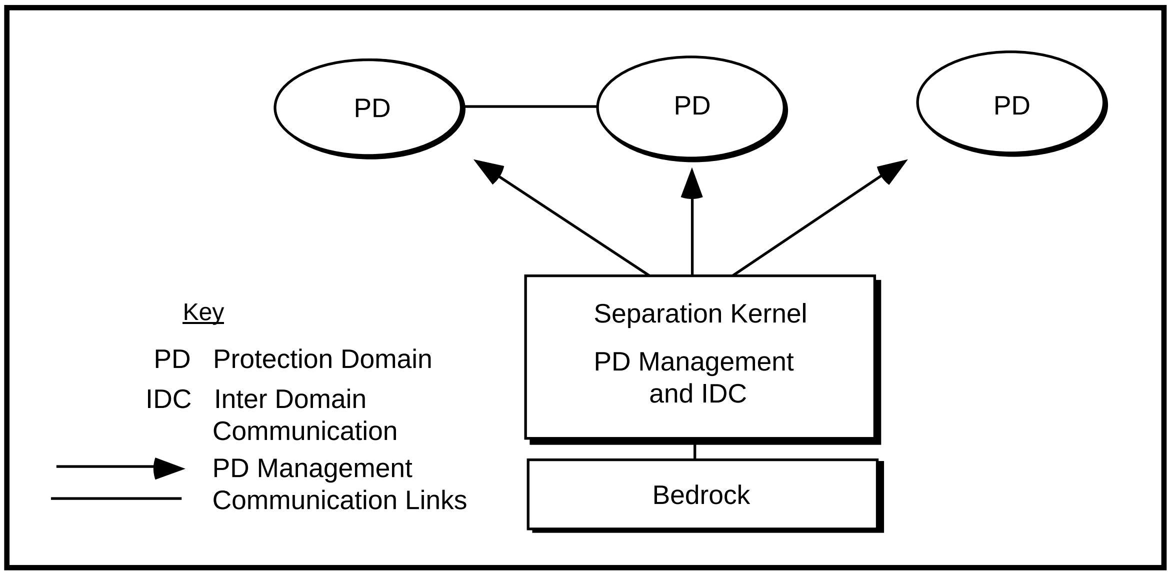

Figure 1. Separation Kernel as a Foundation

1.2 INTERDOMAIN COMMUNICATIONS

There is a need for a mechanism that allows one protection domain to access another protection domain in a controlled manner. For example, a protection domain containing an operating system or a Policy-Enforcing Application needs to be able to receive service requests from a user application and reply to those service requests. The mechanism overseeing these interactions and the criteria on which the access decision is made must allow flexibility and not introduce flaws that compromise the integrity or trustworthiness of the protection domains themselves.

Subject and object attribute-based access control criteria are not appropriate for controlling interaction between protection domains. Protection domains are not objects and they are not entities in the logical address space. Domain attributes provide the basis for controlling the interactions between protection domains. The security kernel associates these attributes with a protection domain when the protection domain is created. The domain attributes are subsequently employed as the basis for validating interdomain communications.

The allowed communications links between protection domains are defined by a Security Domain Administrator in an Interdomain Communications Policy. If the domains come under the authority of different Security Domain Administrators, an understanding between Security Domain Administrators is necessary for the establishment of the Interdomain Communications Policy. The policy could specify that the protection domains are to be completely disjoint with no connections allowed. On the other hand, the Interdomain Communications Policy might specify that connections are allowed but that there are some restrictions on the connections. The rules implemented in most Interdomain Communications Polices will probably reflect a middle ground with some restrictions on the connections between protection domains.

An important characteristic of this approach is that the domain-to-domain Access Control Policy is independent of any Access Control Policy used to regulate access to objects within a domain. The separation kernel manages protection domains. The separation kernel is not concerned with specific functions within a protection domain.

The extreme cases of Interdomain Communications Policy require no special implementation mechanism. Complete isolation implies absence of all mechanisms. Free and open communications are accomplished using normal communications mechanisms, but not security-relevant functions.

When protection domains are implemented on separate processor bedrock, the interdomain communications remains. Its implementation may become more complex as the interdomain communications paths become longer, more exposed to external influence, and of less bandwidth than on a single platform. Perhaps the separation kernel should be renamed the interdomain communications kernel when there are separate hardware domains.

A few examples of domain communications are examined to illustrate interdomain communication alternatives and issues. The examples begin by examining the use of a local device. These are services similar to the input/output functions provided by a traditional operating system. The subsequent examples examine interdomain communications issues.

1.2.1 Communication with a Device

In the first example, the protection domain created by the separation kernel is isolated from other protection domains but is provided with access to an external device (see Figure 2). This device could be a disk device, a network communications device, or any other bedrock resource. Within this protection domain, an input/output (I/O) function controls the flow of data between the protection domain and the device. Access to this I/O function may be mediated by a security function that implements an Access Control Policy for operations on the device.

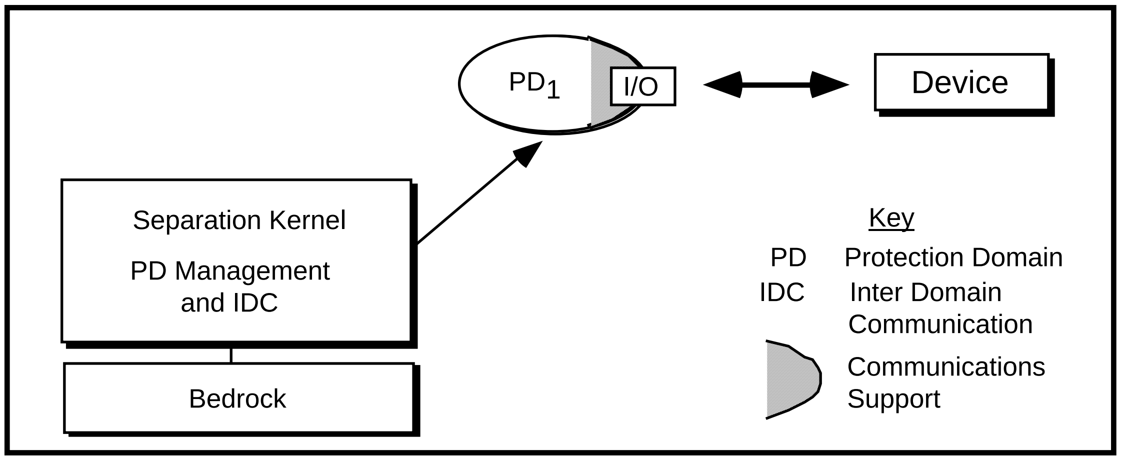

Figure 2. Protection Domain with a Device

1.2.2 Direct Communication Between Two Protection Domains

The second example involves communication between two protection domains (see Figure 3). In this example, the Interdomain Communications Policy allows communication between protection domain$_1$ and protection domain$_2$. The specific restrictions on domain-to-domain communications are specified in the Interdomain Communications Policy. Both domains contain input/output functions that control the flow of data between the protection domains, and perhaps to external devices or other protection domains. Depending upon the complexity or sophistication of the communications services, these interdomain communications functions might contain a protocol stack that modulates the flow of information, performs error correction, or provides high-level logical services.

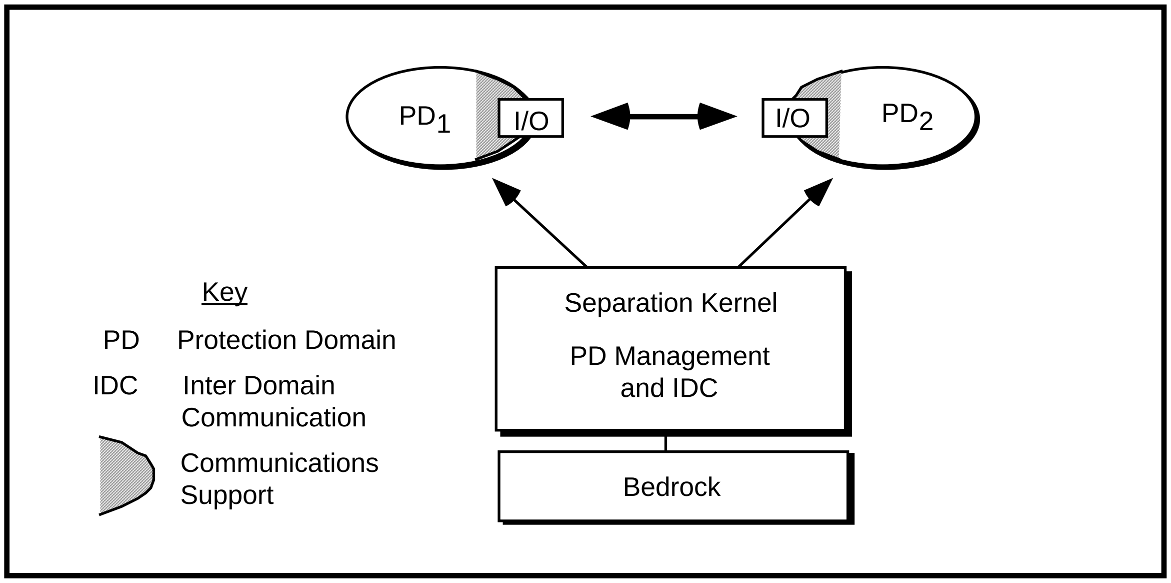

Figure 3. Communication Between Two Protection Domains

In this example, enforcement of the Interdomain Communications Policy is allocated to instances of the Reference Validation Mechanism in both domains. This is similar to the partitioned Trusted Computing Base described in the *Trusted Network Interpretation* in which the Network Trusted Computing Base partition in a subsystem enforces device range restrictions on data exchanged with another subsystem. There are two policies at work here: the interdomain communications policy controls establishing the link. Once the link is established, the application in the protection domain implements an application specific policy.

1.2.3 Mediated Communication Between Protection Domains

The third example involves mediated communication between protection domains (see Figure 4). In this example, the Interdomain Communications Policy does not allow direct communication between protection domains protection domain$_1$ and protection domain$_3$. Therefore, an intermediary protection domain, protection domain$_2$, serves as the connection between the domains. In this situation, a security-enforcing function called a *Gate Keeper* is introduced to manage the flow of information between the protection domains. Gate Keepers are logical functions, located in separate protection domains, that facilitate and supervise the flow of information between protection domains according to the Interdomain Communications Policy. Our Gate Keepers are patterned after the similarly named hierarchical protection domain ring mechanism in Multics (Organick, 1972).

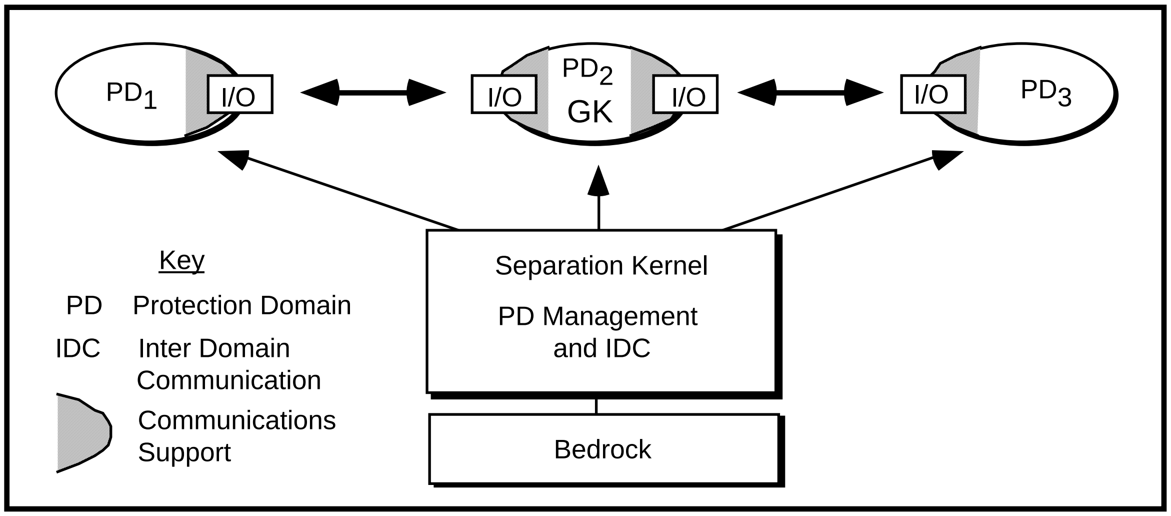

Figure 4. Intermediary Protection Domain with Gate Keeper

To achieve security goals of high-assurance systems, the Gate Keeper approach provides a convenient mechanism for addressing covert channel requirements. When the overall system security policy includes a covert channel policy, the Gate Keepers can be used to enforce the covert channel policy. The Gate Keepers, for example, may audit or restrict the bandwidth of identified covert channels. As with Multics, the Gate Keepers may validate the arguments and parameters in the information flow between protection domains. Mediation efficiency and efficacy may be enhanced by the use of a limited variety of fixed-format messages. Since this communications path is not intended for human communication, fixed-format messages pose no hardship.

1.3 ACCESS CONTROL

A separation kernel provides the rigorous isolation needed in an Information Technology system to gain properties such as integrity, confidentiality, and Trusted Computing Base self-protection. However, a modern Information Technology system must do more than protect itself; it must support a variety of access control needs, including those required to satisfy local needs.

Breaking down a complex problem into smaller, more understandable pieces has also been applied with success to the access control process. From the monolithic approach of early operating systems, the access control process has been examined and divided into more fine-gained components and access control information. The Extended Access Control Framework presented in the second paper, for example, identifies both elemental functions and information required to support those functions. It also explicitly addresses the presence of multiple Access Control Polices.

The Generalized Trusted Computing Base integrates the characteristics of the separation kernel with desirable access control features, such as those offered by the Extended Access Control Framework. This concept separates the access control computation from the operational computation performed in support of the object management function. The operational computation is concerned with providing some specific object management service. The goal of the access control computation is mediation of access to objects. For example, in a Policy-Enforcing Application such as a database management system, the distinction is between data management functions and access control functions.

The modularity provided by the separation kernel is a key to meeting the wide diversity of access control needs. Analysis begins with determining the influence that the availability of protection domains will have on the architecture of the system. The protection domain framework encourages composing an application from small components and placing those components into separate protection domains. Functional decomposition is a typical approach, although other approaches have been used with success. Whatever the specific approach used, dividing a system into smaller pieces promotes understandability by allowing a system to be built from constituent pieces that are small enough to be subjected to analysis and tests to assure that they are correct.

This high-level separation of functions can be continued by identifying and separating specific functions within the access control process. Access control, especially in the presence of multiple Access Control Policies, is difficult to understand and complex to implement. Using the Access Control Framework as a reference, the overall access control process can be divided into individual sets of Access Control Rules implemented in separate Access Control Decision Functions, an Access Enforcement Function, and a Metapolicy Function to combine the votes of the individual Access Control Decision Functions. This approach provides a detailed and understandable view of the access control process and, in a multiple-policy environment, helps highlight the interaction of the individual Access Control Policies.

There are several advantages of this separation. Separation of these functions prevents the accidental intermingling of information requiring different forms of protection, prevents corruption of the access control mechanism, and explicitly identifies the access control functions.

Using the isolation provided by the separation kernel or by separate hardware bedrock, access control functions can be distributed into separate protection domains (Figure 5). Figure 5 collapses the detail of the gatekeepers. In this example, each individual set of Access Control Rules is implemented in a separate Access Control Decision Function and is located in separate protection domains. This process protects each Access Control Decision Function and also allows maintenance of individual Access Control Decision Functions without a major impact on the Trusted Computing Base within a protection domain. Although the change or creation of the rules implemented in an Access Control Decision Function is not to be undertaken lightly, a change within this type of framework is much less involved than the change or creation of a trusted product. Moreover, the change is well defined and localized.

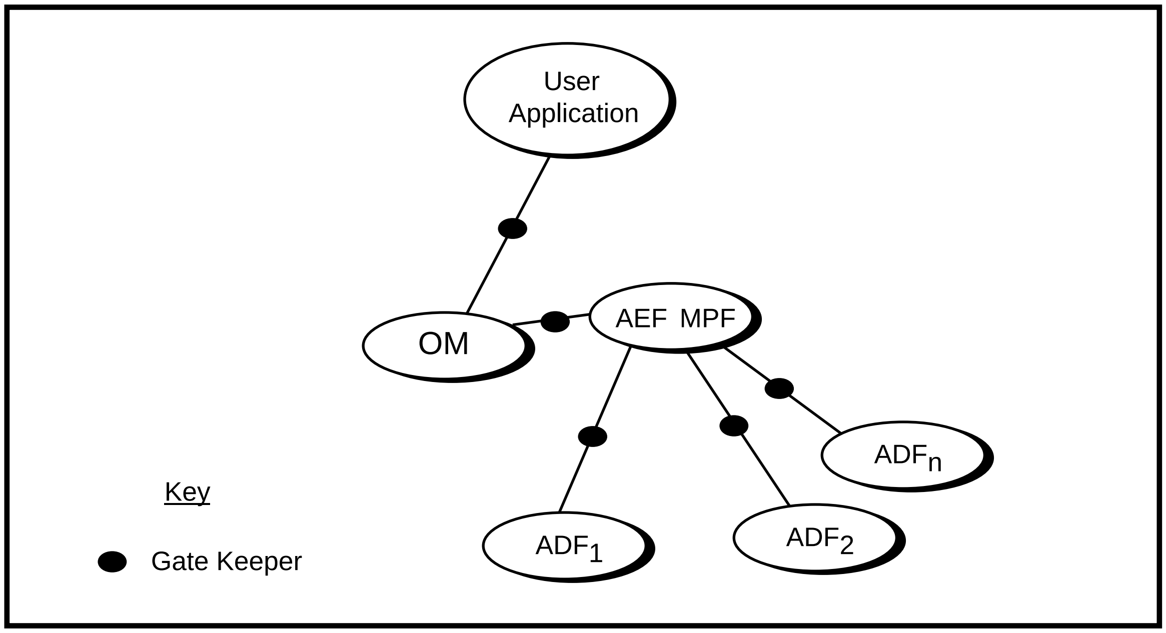

Figure 5. Protection Domains and Access Control Functions

The Access Enforcement Function and the Metapolicy Function, shown in Figure 5, are located in another protection domain. There is a close coupling of the Access Control Enforcement Function and Metapolicy Function, the Metapolicy Function combines the votes of the various Access Control Decision Functions into a single vote passed to the Access Control Enforcement Function, and it seems logical that these functions could be combined into a single protection domain. This scheme also can localize the changes when there is interest in introducing additional Access Control Policies or changing the metapolicy relationship among the constituent Access Control Policies.

Gate Keepers are included on each communications link in Figure 5. The Gate Keepers facilitate and supervise the flow of information between protection domains and implement the required Interdomain Communications Policy. The Interdomain Communications Policy, for instance, might require minimal Gate Keeper functionality on the links between the protection domains containing the Access Control Decision Functions and the domain containing the Access Control Enforcement Function and Metapolicy Function. On the other hand, the Interdomain Communications Policy might require a more complex Gate Keeper on the communications link between the application program and the Object Manager. This interdomain connection might involve close scrutiny, including the use of auditing or bandwidth restrictions to satisfy covert channel requirements.

1.4 RELATIONSHIP BETWEEN POLICY COMPONENTS

The Generalized Trusted Computing Base concepts identifies two major policy components. Figure 6 illustrates the relationship between the Interdomain Communications Policy and the Access Control Policies. Each Protection Domain contains a Policy-Enforcing Application that creates objects and implements its own unique Access Control Policy. The policy within a domain is expressed as a set of rules and is enforced by the Reference Validation Mechanism within the application. The Interdomain Communications Policy defines allowed domain-to-domain communications links.

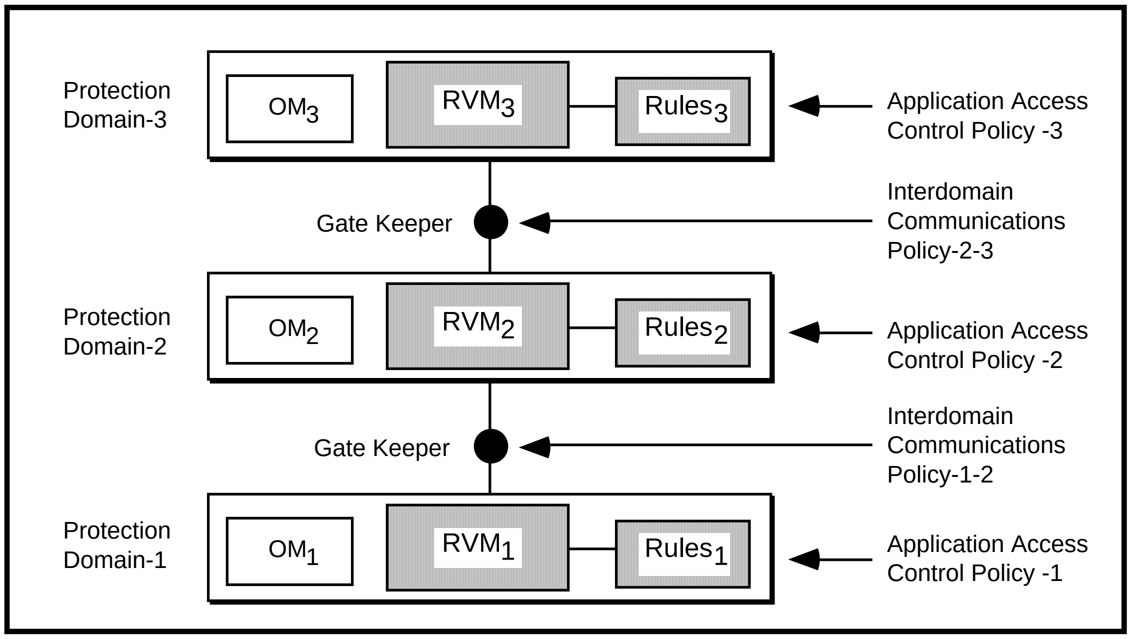
Protection Domain-3
OM3
RVM3
Rules3
Application Access Control Policy -3
Gate Keeper
Interdomain Communications Policy-2-3
Protection Domain-2
OM2
RVM2
Rules2
Application Access Control Policy -2
Gate Keeper
Interdomain Communications Policy-1-2
Protection Domain-1
OM1
RVM1
Rules1
Application Access Control Policy -1

Figure 6. Interdomain Communications Policy and Access Control Polices

With distributed systems or Policy-Enforcing Applications, the Security Domain Attributes for each domain may be different. Within each separation domain, the Security Domain Administrator is the ultimate authority, but not necessarily between domains. The Interdomain Communications Policy between domains i and j, identified as Interdomain Communications Policy$_{ij}$, must be agreeable to Security Domain Administrator$_i$ and Security Domain Administrator$_j$. If they are both subordinate to a higher Security Domain Administrator$_H$ and cannot agree, then Security Domain Administrator$_H$ can set the Interdomain Communications Policy. But if they are not compatible (e.g., they belong to different nations), then they must come to an agreement in establishing Interdomain Communications Policy$_{ij}$.

1.5 RELATIONSHIP TO PRIOR IMPLEMENTATIONS

The Generalized Trusted Computing Base builds on the concepts of a separation kernel and the separation of access control functions. The applicability of using a separation kernel, also called a *virtual machine monitor*, to create multiple instances of an operating system for security purposes has received some attention (Kelem, 1991). Other variations of this scheme, differentiated primarily by the amount of resource sharing the kernel allows, have been proposed for several general-purpose computer systems (Russell, 1989) and implemented by the Digital Equipment Corporation (Karger, 1990), in KVM/370 (Gold, 1984), and most recently by Amdahl (Graff, 1992). Figure 7 illustrates the Amdahl approach employing a separation machine, which creates individual environments. Each environment contains a System Control Program, which is an instance of an operating system. The flow of information between System Control Programs is controlled by high-level processes called *guards*. The Figure suggests a brick wall separating the domains.

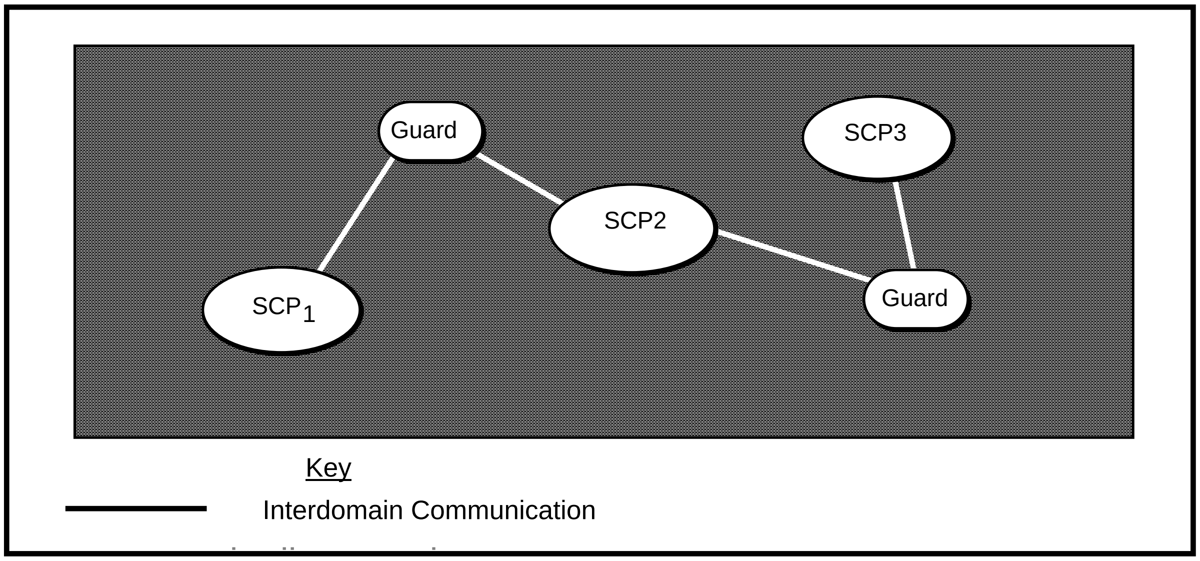

Figure 7. Amdahl Separation Machine

This approach represents an implementation that closely parallels the Generalized Trusted Computing Base concept. The Separation Machine and environments correspond to the separation kernel and protection domains, respectively, in the Generalized Trusted Computing Base Subsets concept.

2 BENEFITS

The Generalized Trusted Computing Base offers a new and flexible framework for understanding the relationships among the building blocks of contemporary trusted systems. The bedrock provides the resources upon which the system can be built. The separation kernel transforms these resources into protection domains. Protection domains provide the rigorous isolation and separation in which to construct Trusted Computing Bases and user processes. The creation and management of protection domains are the only functions of the separation kernel.

With isolation and separation ensured by the separation kernel or separate hardware bedrock, controlled communication between protection domains is provided through the Gate Keepers. Gate Keepers facilitate and supervise the information flow between protection domains. The allowed flow of information between protection domains is determined by an Interdomain Communications Policy.

Access control issues have also been addressed. Policy-Enforcing Application are becoming the principal building blocks in a trusted system. In addition to their object management function, Policy-Enforcing Applications contain an access control function. The Generalize Trusted Computing Base concept allows separation of the access control computation from the operational computation performed in support of the object management function.

2.1 STANDARDIZATION OF ACCESS CONTROL POLICY MODULES

Identification and separation of access control functions allows the investigation of standardization of access control requirements. The Generalized Trusted Computing Base emphasizes separation of the access control rules from the access control enforcement mechanisms. Assuming an implementation that parallels this framework, this approach offers the potential for implementing generic, approved, commercial off-the-shelf Access Control Policy enforcement modules that can be combined with approved modules containing specific access control rules. The Access Control Enforcement Function, Metapolicy Function, and Access Control Decision Function mechanisms, for instance, can be evaluated and approved by a suitable authority independent of the Access Control Policies expressed in the Metapolicy Rule and Access Control Rule.

Identification and isolation of specific rules sets has the advantage of allowing specific policies to be evaluated and approved by authorities for those rules. For example, in the commercial environment, we envision professional organizations becoming the source of approved Access Control Policy modules in such industry segments as banking, insurance, and health services. High reliability would be an outgrowth of this approach. Once approved, these modules could be integrated with the approved enforcement mechanism.

2.2 INCREASED PERFORMANCE

The Generalized Trusted Computing Base concept, coupled with the Access Control Enforcement Function-Metapolicy Function-Access Control Decision Function reformulation of the Reference Monitor, also opens the possibility for improved performance of trusted Information Technology systems. Depending upon the capabilities of the supporting hardware systems, the separation of individual Access Control Policies into separate Access Control Decision Functions allows for parallel adjudication of the Access Control Policies by the Access Control Decision Functions.

Hardware support of the individual protection policies can further improve performance. We observe a trend in the evolution of computer system architecture whereby new concepts are first introduced in software. The software and the concepts implemented are iterated until stability is achieved. If the concepts prove valuable, they achieve a degree of use that begins to degrade system performance. This performance degradation is often offset by adding hardware to the system configuration. This additional hardware may provide an increase in the computational throughput rate. Hardware floating point coprocessors are a well-known example. The LOCK (Boebert, 1988) system, which employs a coprocessor that plugs into a backplane to adjudicate access, is another example of hardware support for access control.

Developing this trend to implement access control functions in hardware, Figure 8 pictures hardware and firmware that support the concepts discussed in this paper. The board on the left implements the separation kernel. The enforcement functions may be implemented in integrated circuits or firmware. The rules for interdomain communication and multipolicy adjudication will most likely be stored in firmware. The board on the right implements the Access Control Policy within a protection domain. It includes the enforcement function in hardware or firmware and the rules in firmware.

2.3 IMPROVE THE ABILITY TO CHANGE ACCESS CONTROL POLICIES

One of the potential strengths of an implementation based upon the Generalized Trusted Computing Base is that it provides a straightforward way to implement policy changes. In the current environment, the Access Control Policy enforced by the Information Technology system is non-extensible. That is, the Access Control Policy implemented by the developer of the Information Technology system can neither be modified nor extended. Consequently, changes or additions to this policy are not easily handled, especially if those changes occur after delivery of the Information Technology system.

Furthermore, in the current environment the relationship among the constituent Access Control Policies is based upon the logical AND operation. That is, an attempted access request must pass the access control criteria specified by all of the Access Control Policies before access is granted. As discussed by Abrams (1993), the votes of all of the Access Control Decision Functions must be combined to adjudicate the access request. However, the operation used to combine the Access Control Decision Function votes does not necessarily have to be the logical AND operation. The rules (i.e., Metapolicy Rule) supplied to the Metapolicy Function offer the flexibility to implement arbitrarily complex combinations.

In the Generalized Trusted Computing Base environment, changes in Access Control Policy can be implemented by composing and substituting a new set of Access Control Rules and Metapolicy Rules. The change or creation of rules is not to be undertaken lightly, but is much less involved than the change or creation of a trusted product. Hence, the evaluation of a new or changed rule set should be less complex and less time consuming than a product evaluation.

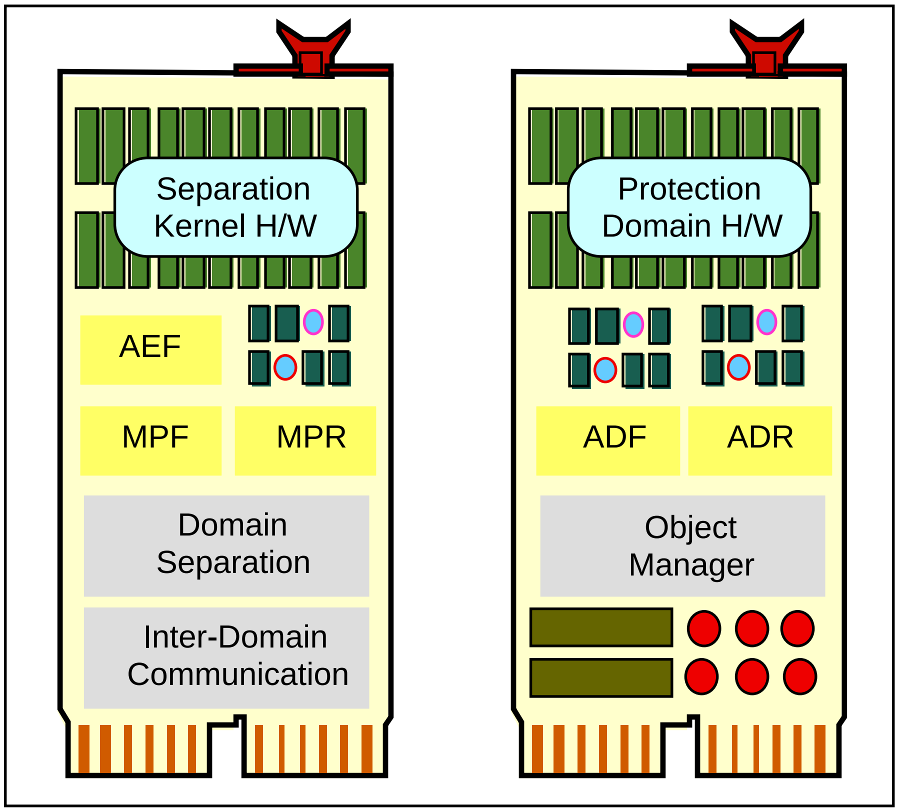

Figure 8. Hardware Support for Separation Kernel and Protection Domains

2.4 IMPROVE THE ABILITY TO INTEGRATE TRUSTED PRODUCTS

The Generalized Trusted Computing Base concept assumes that the system is composed of a collection of protection domains. A protection domain might contain an operating system and/or one or more Policy-Enforcing Applications. Once evaluated, the protection domain becomes a Trusted Computing Base Subset that could be used as a resource by any other Trusted Computing Base Subset in the same way that the hardware and firmware are used as a resource by a traditional, non-subsetted Trusted Computing Base. The motivation is to employ any existing, evaluated Trusted Computing Base Subset as a building block on which to place a new Trusted Computing Base Subset that provides a useful service. This useful service could be a combination of new Access Control Policies and new objects, typically offering new abstractions of granularity or structure not previously available.

Considerable flexibility is achieved if each Trusted Computing Base Subset can be independently evaluated. Once evaluated, these independently evaluated subsets can be composed into larger modules, subsystems, or even systems without violating the conditions of prior evaluations. This is a strategy for composability and evaluation by parts, as described in the *Trusted Database Interpretation*. One *Trusted Database Interpretation* motivation was the desire to implement trusted Database Management Systems on previously evaluated operating systems without affecting the previously evaluated operating system Trusted Computing Base in any way.

Once a subset is integrated into the system, the allowed interactions between protection domains are determined by the Interdomain Communications Policy implemented in the Gate Keepers. The degree of interaction permitted by the Interdomain Communications Policy would affect the amount of work required to perform an incremental evaluation when a new Policy-Enforcing Application is added to a previously evaluated system. The dependency analysis concept introduced in the Federal Criteria (NIST, 1992) might help identify the areas in which analysis was required.

A restrictive Interdomain Communications Policy would only allow interactions between protection domains containing user applications and the protection domain containing the operating system. Other Interdomain Communications Policies might allow interaction among selected Policy-Enforcing Applications while excluding user applications from these exchanges.

2.5 SUPPORT NEW IMPLEMENTATION STRATEGIES

As mentioned earlier, every constituent Access Control Policy does not necessarily apply to every object. The Unconstrained Data Items in the Clark-Wilson integrity policy and the Organization Controlled policy (Abrams, 1991) are two examples of these types of exceptions.

One approach to identifying the Access Control Policy applicable to an object is to associate a set of security attributes with each subject and object entity (see Figure 9). In this approach, at least one security attribute supports each Access Control Policy, or perhaps a closely related set of Access Control Policies, such as the clearance and classification attributes that support Mandatory Access Control. When an object is created by an object manager in a Trusted Computing Base, a security attribute is associated with the object that indicates that the Trusted Computing Base needs to be involved in an access control decision.

A special value of an attribute, such as null, would indicate that the corresponding Access Control Policy did not apply to that entity. The special security attribute “Public Object” has been applied to the system clock to permit reading by all subjects, yet satisfying Mandatory Access Control and Discretionary Access Control restrictions (NCSC, 1989). In the prototype implementation of the Organization Controlled policy, a Mandatory Access Control category was used to indicate whether the Organization Controlled policy applied to an object.

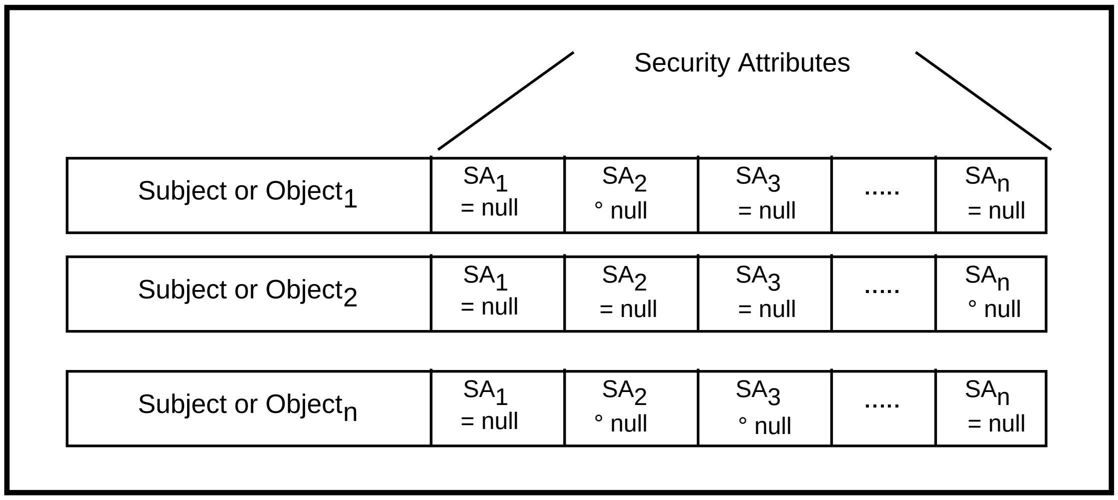

Figure 9. Security Attributes

In making an access control decision, the Access Control Enforcement Function obtains a composite vote from the Metapolicy Function; the Metapolicy Function combines the votes from the individual Access Control Decision Functions using the rules encoded in the Metapolicy Rule. A polling protocol might be employed by the Metapolicy Function to determine the votes of the individual Access Control Decision Functions.

2.6 STRONG ISOLATION FOR ALL COMPONENTS

The Access Control Policy adjudicated by the Access Control Decision Functions and enforced by the Access Control Enforcement Function determines whether a process outside a Trusted Computing Base Subset can access an object. In addition to objects that are accessible, the Trusted Computing Base Subset probably will have internal data structures (i.e., internal objects) built from the same resources that are used to build objects. These data structures are not accessible; they are instances of data hiding. The integrity of the data within these data structures is critical to the correct operation of the Trusted Computing Base Subsets. This integrity is ensured by the Protection Domains created by the separation kernel.

2.7 STRUCTURE TO DEAL WITH COMPLEXITY

Trusted systems can range widely in complexity. Factors influencing the complexity include policies enforced, system architecture, physical environment and distribution, and applications implemented. The Generalized Trusted Computing Base concept employs the time-tested strategy of separating a complex problem into constituent parts with well-defined interfaces between the parts. The authors of the TNI became convinced of the value of this approach as applied to layered communications architectures. Note, however, that the structures described in this paper are more general than strict layers.

Lest readers be put off by apparent complexity, we hasten to point out that not all applications need to be policy enforcing. The Reference Validation Mechanism design puts all trust in the Trusted Computing Base. Generally, applications are outside the Trusted Computing Base and are untrusted. However, the addition of applications, such as database management systems, that manage their own objects and enforce their own policy introduces complexity to the overall architecture as well as to the Trusted Computing Base.

Choice among alternative policies can have a dramatic influence on complexity. For example, consider the two integrity policies developed by Biba and Clark-Wilson. Sandhu (1989) discusses an interesting comparison of these policies. Biba integrity policy employs labels and dominance relations to enforce what it terms *mandatory integrity*. All trust is concentrated in the Trusted Computing Base. The Clark-Wilson integrity policy is more complex; it requires that some applications must be trusted.

The Clark-Wilson model of integrity builds on the idea that one should identify and label those data items within a system to which the integrity model will be applied. Clark and Wilson call these *Constrained Data Items*. A particular integrity policy for a system is defined by two classes of procedures: Integrity Verification Procedures and Transformation Procedures. Both procedures must be part of the Trusted Computing Base. The purpose of the Integrity Verification Procedures is to confirm that Constrained Data Items in the system conform to the system's integrity specification at the time the Integrity Verification Procedures are executed. The purpose of the Transformation Procedures is to change a set of Constrained Data Items from one valid state to another.

To maintain the integrity of Constrained Data Items, a system must ensure that only a Transformation Procedure can manipulate the Constrained Data Items. While the system can ensure that only Transformation Procedures manipulate Constrained Data Items, it cannot ensure that a Transformation Procedure performs a well-formed transformation. The validity of a Transformation Procedure (or an Integrity Verification Procedure) can be determined only by certifying it with respect to a specific integrity policy.

Complexity, policy and application selection, and distribution are part of the problem space in which the system architect works. This paper provides a conceptual framework to better deal with security concerns in that problem space.

2.8 FOCUS FOR IMPLEMENTATION DECISIONS

The Generalized Trusted Computing Base concept assumes an environment in which the building blocks are Policy-Enforcing Applications. However, this concept introduces the issue of how to deal with the relationship between a Policy-Enforcing Application and the operating system, and the relationships among Policy-Enforcing Applications. The Generalized Trusted Computing Base concept allows several alternatives to be considered.

The Access Control Framework provides an expanded view of the access control process to deal with the presence of multiple Access Control Policies. This section identifies design alternatives that might require an Metapolicy Function within a protection domain as well as between protection domains.

The Access Control Policy enforced by a Policy-Enforcing Application can be complex. A clear understanding of the Access Control Policy implemented in a Policy-Enforcing Application is required before it can be integrated with the operating system and other Policy-Enforcing Applications in a trusted manner. When the Policy-Enforcing Application enforces a complex combination of policies, there may be a need for a Metapolicy Function and Metapolicy Rules within the Policy-Enforcing Application protection domain.

Multiple Object Managers within a protection domain strongly suggest the need for multiple Access Control Decision Functions. Following the example of the *Trusted Network Interpretation,* we suggest that an Access Control Decision Function could be associated in a one-to-one relationship with each Object Manager. This relationship might reflect a need for efficiency by making decisions based on local knowledge.[1] While there are multiple Access Control Decision Functions in this situation, only one Access Control Enforcement Function might be used per separation kernel.

The presence of multiple Policy-Enforcing Applications indicates the need for an Metapolicy Function to combine more than one Access Control Policy to obtain the overall system security policy. Each access request for an object is validated against the Application Access Control Policy which is a composite of several constituent Access Control Policies. Each of

[1] This issue was discussed in the *Trusted Network Interpretation* in the following sections: I.3.2.2.1., “Connection-Oriented Abstraction”; B.4.1., “Introduction to the Partitioned NTCB Concept”; B.4.4., “Characterization of the Loosely-Coupled Trusted Network”; B.4.7., “Conclusions Regarding the Simulation Argument”; and B.7.3., “NTCB Partitions.”

these Access Control Policies is implemented within a separate Access Control Decision Function. The Metapolicy Function combines the votes in accordance with the rules implemented in the Metapolicy Rule. The outcome of this process determines whether the user process is allowed to access the objects managed by the application.

However, the framework provided by the Generalized Trusted Computing Base concept can be applied to other architectural alternatives. We recognize that none of these designs are canonical and offer them as examples. Perhaps subsequent researchers and designers will find superior architectures. There is a parallel with the Anderson Report, which presented the Reference Monitor concept as *a* design for building trusted systems. No documentation has been found that explains how the Reference Monitor concept became *the only* acceptable design in the Orange Book.

3 SUMMARY

This set of papers has systematically studied design and composition issues concerned with the implementation of multiple security policies. Building on the Trusted Computing Base Subsets introduced by Shockley and Schell (1987) and refined in the *Trusted Database Interpretation*, we identified a hierarchical relationship of Trusted Computing Base Subsets and offered a more flexible framework based on a separation kernel (Rushby, 1984).

The refinement of the Reference Validation Mechanism concept, based on an Access Control Framework draft ISO standard (ISO, 1993), was employed in the Generalized Framework for Access Control (Abrams, 1990, 1991), and was extended by Abrams (1993) to include a Metapolicy Function and Metapolicy Rules. Both Trusted Computing Base Subsets and the proposed extended ISO Access Control Framework support the implementation of multiple security policies. Trusted Computing Base Subsets allow reuse of existing operating system Trusted Computing Bases as platforms on which to build trusted Policy-Enforcing Applications such as database management systems. New designs, models, and implementation methods should be patterned after this approach because of the simplicity and flexibility it offers.

Both generalized Trusted Computing Base Subsets and the proposed extended ISO Access Control Framework also support evaluation by parts. The Access Control Enforcement Function, Metapolicy Function, and Access Control Decision Function will only have to be evaluated once. The Metapolicy Rules and Access Control Rules can be evaluated independently. The evaluation of a new or changed rule set should be less complex and less time-consuming than a product evaluation.

The concept of Bedrock presented a new view of the conventional resources of hardware, software, and firmware on which a Trusted Computing Base is constructed. The discussion about Bedrock reminds us not to place undue trust in technology. The objects made available by a Trusted Computing Base are equivalent to Bedrock for another Trusted Computing Base that employs these objects as resources.

ACKNOWLEDGMENTS

The authors wish to acknowledge the ideas and encouragement received from the anonymous reviewers and colleagues who read prior drafts of this paper: Ed Amoroso, Blaine Burnham, David Gomberg, Jody Heaney, Ralf Houser, Dale Johnson, Len LaPadula, Carol Oakes, and Jim Williams. This work was supported by the National Security Agency under contract DAAB07-93-C-N651 and by the MITRE Corporation.

LIST OF REFERENCES

Abrams, M. D., K. W. Eggers, L. J. LaPadula, and I. M. Olson, October 1990, “Generalized Framework for Access Control: An Informal Description,” *Proceedings of the 13th National Computer Security Conference.*

Abrams, M. D., J. Heaney, O. King, L. J. LaPadula, M. Lazear, and I. M. Olson, October 1991, “Generalized Framework for Access Control: Towards Prototyping the Organization Controlled Policy,” *Proceedings of the 14th National Computer Security Conference.*

Abrams, M. D., M. V. Joyce, and J. E. Heaney, October 1992, “Mediation and Separation in Contemporary Information Technology Systems,” *Proceedings of the 15th National Computer Security Conference.*

Abrams, M. D., and M. V. Joyce, 1993, “Extending the ISO Access Control Framework for Multiple Policies,” *Proceedings of the Ninth International Information Security Conference*, Elsevier Science Publishers.

Boebert, W. E., 1988, “The LOCK Demonstration,” *Proceedings of the 11th National Computer Security Conference*, Baltimore, MD.

Department of Defense, 1985, *Department of Defense Trusted Computer System Evaluation Criteria*, Department of Defense 5200.28-STD, Washington, DC.

Gold, B. D., et al., 1984, “KVM/370 in Retrospect,” *Proceedings of the 1984 IEEE*

Symposium on Security and Privacy, Oakland, CA, IEEE Computer Society Press.

Graff, J., 1992, “Separation Machines,” *Proceedings of the 15th National Computer Security Conference*, pp. 631-640.

International Organization for Standardization (ISO), September 1993, International Electrotechnical Committee, Joint Technical Committee 1, Subcommittee 21, *Information Technology - Open Systems Interconnection - Security Frameworks in Open Systems - Part 3: Access Control*, Document Number ISO/IEC JTC/SC 21 N 8224 (or most recent draft).

Karger, P. A., et al., 1990, “A VMM Security Kernel for the VAX Architecture,” *Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy*, Oakland, CA, IEEE Computer Society Press, pp. 2-19.

Kelem, N. L., and R. J. Feiertag, 1991, “A Separation Model for Virtual Machine Monitors,” *Proceedings of the 1991 IEEE Symposium on Research in Security and Privacy*, Oakland, CA, IEEE Computer Society Press, pp. 78-86.

National Computer Security Center (NCSC), 9 January 1989, *Discretionary Access Control Public Objects Interpretation*, Report No. C1-CI-03-89.

National Institute of Standards and Technology (NIST) and National Security Agency (NSA), December 1992, *Federal Criteria for Information Technology Security*, draft.

Organick, E., 1972, *The Multics System: An Examination of its Structure*, M.I.T. Press.

Rushby, J., September 1984, “A Trusted Computing Base for Embedded Systems,” Proceedings *of the 7th Department of Defense/NBS Computer Security Conference*, pp. 294-311.

Russell, T. T., and M. Schaefer, 1989, “Toward a High B Level Security Architecture for the IBM ES/3090 *Processor Resource/Systems Manager,” Proceedings of the 12th National Computer Security Conference*, pp. 184-196.

Sandhu, R. S., December 1989, “A Perspective on Integrity Mechanisms,” *Proceedings of the Fifth Annual Computer Security Applications Conference*, Tucson, Arizona, page 279.

Shockley, W. R., and R. R. Schell, December 1987, “Trusted Computing Base Subsets for Incremental Evaluation,” *Proceedings of the Third Aerospace Computer Security Conference*, American Institute of Aeronautics and Astronautics, CP 8711.